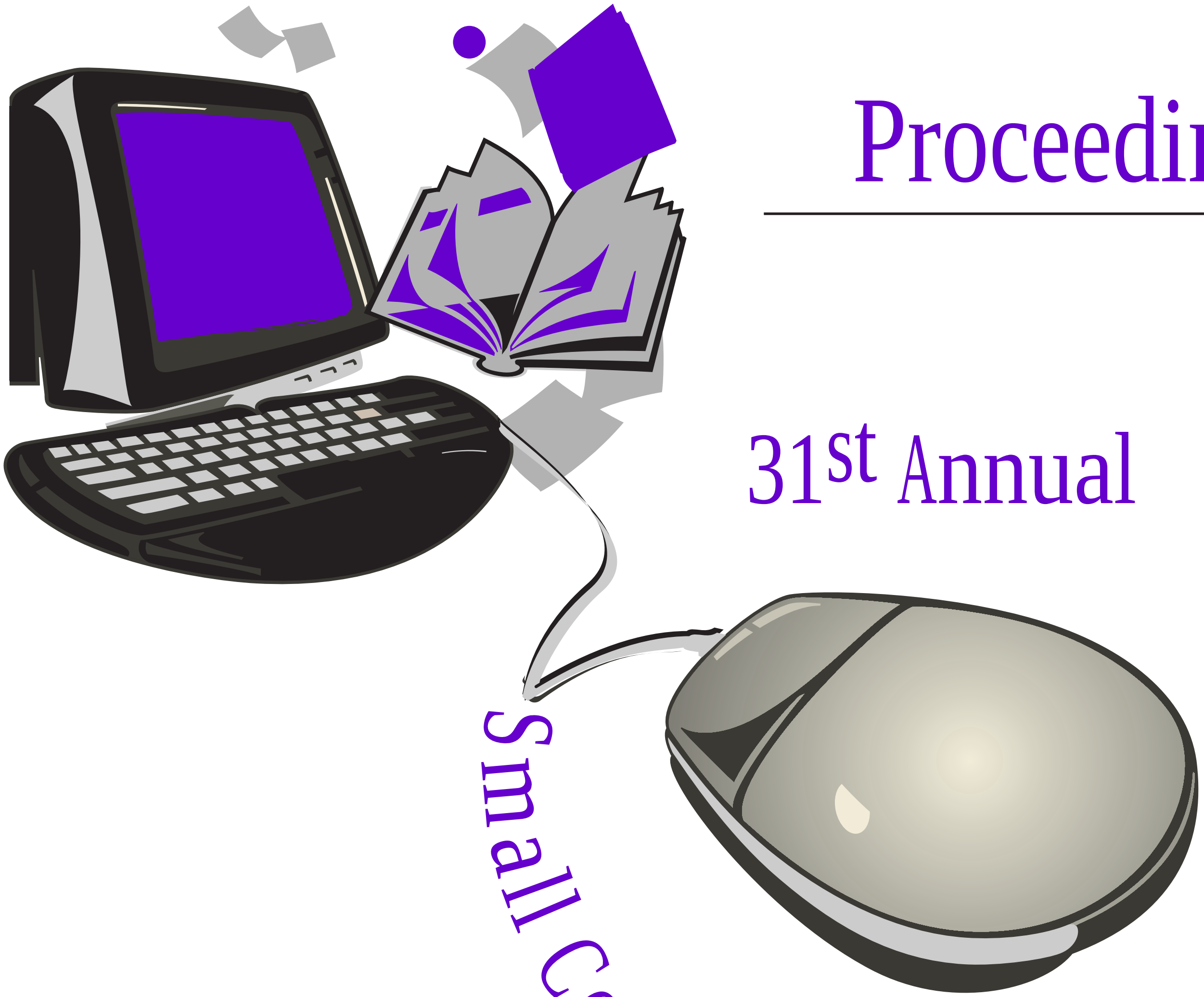
SCCS '98

Building the Virtual Campus: Measuring Our Progress



Proceedings

31st Annual

Small College Computing Symposium

Sponsored by:

Concordia College - Moorhead, Minnesota

Moorhead State University - Moorhead, Minnesota

North Dakota State University - Fargo, North Dakota

Northwest Technical College - Moorhead, Minnesota

April 16-18, 1998

SCCS

PROCEEDINGS

31st Annual
Small College Computing Symposium

Building the Virtual Campus: Measuring Our Progress

April 16 – 18, 1998

Concordia College – Moorhead, Minnesota
Moorhead State University – Moorhead, Minnesota
North Dakota State University – Fargo, North Dakota
Northwest Technical College – Moorhead, Minnesota

Preface

The annual Small College Computing Symposium (SCCS) is a regional forum for the-discussion of education activities and opportunities involving the use of computer related technology. Conceived by Conrad Dietz and Gene Kemper of the University of North Dakota in 1968, it was an effort to get educational users of computers to congregate and share experiences, successes, and failure so as to promote high-quality computing at institutions of higher learning. Since that time, it has grown into an annual spring event attended by hundreds of faculty and staff from colleges and universities nationwide.

Continuity of the symposium is assured by a Steering Committee. A goal of the Steering Committee is to promote quality symposia at affordable cost. Accordingly, the committee carefully selects host institutions, provides guidance and assistance to host chairpersons, and maintains historical records, a mailing list, and a repository of past conference proceedings. The Steering Committee holds the copyright to the proceedings.

Between symposia, activities of the SCCS, as a legally incorporated not-for-profit organization, are conducted by the Steering Committee. Although program planning and financial arrangements for a symposium are appropriately the responsibility of the host institution, the Steering Committee does assist the host chairperson(s) if requested and reviews the institutional financial plan for the symposium. Host institutions assume the responsibility for financial lasses and agree to relinquish any excess revenue to the Steering Committee. Participation of the Steering Committee members is contributed by their institutions.

SCCS
Proceedings of the 31st Annual Small College Computing Symposium
April 16-18, 1998
North Dakota State University
Fargo, North Dakota

1998 SCCS Steering Committee

Sheila Castaneda
Clarke College

Narayan C. Debnath
Winona State University

J. Philip East
University of Northern Iowa

Susan Gammill (Host Chair)
Concordia College

Stephen Hawk
University of Wisconsin – Parkside

Gene Kemper

Randy Kolb (President)
St. Cloud State University

Andy Lopez
University of Minnesota-Morris

Mary-Alice Muraski (Treasurer)
University of Wisconsin-River Falls

Joe Pagone
Augustana College

Donna Raleigh (Secretary)
University of Wisconsin-Eau Claire

Catherine Roraff
University of Wisconsin-Eau Claire

Jayavel Sounderpandian
Univeristy of Wisconsin-Parkside

Sandy Sprafka (Host Chair)
North Dakota State University

Karen Sutherland
University of Wisconsin-La Crosse

John Wasson (Host Chair)
Moorhead State University

1998 SCCS Coordinating Committee

Susan Gammill
Concordia College

Lisa Nordick **Sandy Sprafka**
North Dakota State University

Les Bakke **John Wasson**
Moorhead State University

Pat Wilber
Northwest Technical College

1998 SCCS Local Committees

Tri-College University

Jean Strandness
Maggie Skrogstad

Concordia

Elizabeth Danielson
Roger Haglund
James Hewitt
Bryan Luther
Molly Pederson
David Sprunger
Alex Sze
Philip Thompsen

North Dakota State University

Ross Collins
Jan Flack
Lonnie Hass
Diane Jackman
Paul Juell
Nancy Lilleberg
Jim Ross
Bernie Saini-Eidukat
Larry Schwartz

Moorhead State University

Jeanne Alm
Les Bakke
Curtis Bring
David Crockett
Carol Dobitz
Gerald Hart
Molly Moore
Lawrence Reed
Mark Strand
Wade Swenson
Dennis Van Berkum

Northwest Technical College

Rick Bjorklund
Mary Henriksen
Dean Hunter
Ron Kivi
Tom Koehnlein
Shiela Lapp
Gayle Melberg
Tim Preuss
Kristin Rahl
Tony Sorum
Dan Sponsler

Index By Title

A Case-Based Design for a Game Player [+]

A Course on Genetic Algorithms for Seniors and Graduate Students [+]

A Pilot Study: Offering a Web Course at a Small, Liberal Arts College [+]

A Study of Techniques for Introducing Concurrency into List Processing Library Routines in Modern Concurrent Functional Programming Languages [+]

A Superdisplay Project [+]

A Team Approach to Multimedia Design, Development, Testing and Evaluation: NDSU's New Center for Academic Information Technology [+]

Academic Planning: IT as a Strategic Asset in Higher Education [+]

Advanced Microcontroller Interfacing Laboratory [+]

Algorithm Animation using Java [+]

Application of Active Networks in Distributed Systems [+]

Beyond a First Course in Windows Programming: Why, How, and What? [+]

Bringing Streaming Media to Campus [+]

Converting a Course's Presentations from Legacy Material to the Web [+]

Deploying Port Manageable Hubs to Create Virtual LAN's to Improve Network Performance in a General Education Computer Lab [+]

Design Patterns in Java: *Sutherland* A Software Framework for Evolutionary Computation [+]

Development of Daily Homework and Quiz Manager (DHQM) for Enhancing Teaching/Learning in Large Classes [+]

Five Exercises for Incorporating Moral and Ethical Issues into the Computer Science Curriculum [+]

Flood! Recovering from a Natural Disaster of Epic Proportions! [+]

From On-Line to On-Task: Communicative Technology in the First Year Classroom [+]

Global Information Society: A Multidisciplinary Course [+]

Impact of Technology on Quantitatively Oriented Business Classes [+]

Integrating Technology into a Computer Information Systems Classroom Instruction—What Works and What Doesn't [+]

Interactive Mathematics Laboratory [+]

Internet Research in the Humanities: Teaching Critical Thinking to First Year Students [+]

Measuring Learning Outcomes in a Competency-Based CS 2 Course [+]

Multi-facet Distance Education Services for Educational Leadership (Panel) [+]

NetApp - A Client/Server Application Builder [+]

Networking Option within the MIS Concentration [+]

Notebook Computers for Students: Planning and Implementation on Two Campuses [+]

Organizing for the University Web Site [+]

Participation in NCSTRL: A Digital Library Experience [+]

Planet Oit: a Virtual Environment and Educational Role-playing Game to Teach Geosciences [+]

Portable Cluster Management (Panel) [+]

Preparing Computer Science Graduates for the PC World [+]

Providing Universal Student Access to Technology: A summary of Alternative Models [+]

Security and Flexibility: Computer Labs in a Small College [+]

Security Policies: What Campus Managers Need to Know [+]

Senior Research Project [+]

Smiley: A Web-based Remote Sensing Data Mining System [+]

Software Engineering Classes Develop an Outcome Assessment Software Tool [+]

Solving the Puzzle of Matching Content to Technology [+]

Something Old, Something New, Something Borrowed, Something Blue in Distance Education [+]

Student Technology Teams (Panel) [+]

Surfin' the Net: A Personal Finance Class Project [+]

Surviving SPSS on the Desktop [+]

Teamwork in Computer Science Classes: Advantages and Problems (Panel) [+]

Technology, Learning, and Equity Issues [+]

Time & Space Independent Learning: On-Line Course Offerings at a Small Liberal Arts College- A Status Report -- [+]
University of North Dakota Academic Affairs Web Initiative Pilot --- [+]
Using Hash Tables to Investigate Zipf's Law --- [+]
Virtual Dragon Days: Web-based Registration and Orientation -- [+]
Virtual Lecture, Virtual Laboratory, or Virtual Lesson? --- [+]
Using Faculty to Provide Technological Leadership and Support: A Critical Review (Panel) ------------------------ [+]
Web and Database Integration for the Delivery of Dynamic Information (on a Budget) ------------------------------ [+]
Year Three of NDSU's Instructional Web Project --- [+]

Table of Contents

KEYNOTE SESSIONS

Academic Planning: IT as a Strategic Asset in Higher Education---------------------------------- [+]
Graves, William
Something Old, Something New, Something Borrowed, Something Blue in Distance Education --- [+]
Gibson, Chere Campbell

CONCURRENT SESSIONS

Security Policies: What Campus Managers Need to Know -- [+]
Abbas, Melanie
Virtual Dragon Days: Web-based Registration and Orientation ------------------------------------ [+]
Alm, Jeanne; Bakke, Les; Grim, Stacy
Participation in NCSTRL: A Digital Library Experience -- [+]
Amundson, Dennis; William Perrizo
Flood! Recovering from a Natural Disaster of Epic Proportions!---------------------------------- [+]
Bornhoeft, Doris; Hilley, Cathy
Notebook Computers for Students: Planning and Implementation on Two Campuses ---------- [+]
Brown, Ray; Brown, Linda; Hagen, Gary; Tykwinski, Joe
Providing Universal Student Access to Technology: A summary of Alternative Models ------- [+]
Brown, Ray; Heeler, Phillip; Von Holzen, Roger
Using Faculty to Provide Technological Leadership and Support: A Critical Review (Panel) - [+]
Covey, Mark; Sullivan-Trainor, Deborah
Software Engineering Classes Develop an Outcome Assessment Software Tool ---------------- [+]
Drake, Janet; VanDaele, Eric
Teamwork in Computer Science Classes: Advantages and Problems (Panel) -------------------- [+]
Drake, Janet; Schnepf, Jim; Sutherland, Karen; Wikstrom, Milton
Solving the Puzzle of Matching Content to Technology -- [+]
Draper, Betsy; Peterson, Denise; Rose, Madeleine
Interactive Mathematics Laboratory--- [+]
Farmer, William
Student Technology Teams (Panel) --- [+]
Ficek, Rhonda; Ross, James; Smith, Elizabeth
Measuring Learning Outcomes in a Competency-Based CS 2 Course ---------------------------- [+]
Ftizgerald, Sue
Deploying Port Manageable Hubs to Create Virtual LAN's to Improve Network Performance in a General Education Computer Lab -- [+]
Guster, Dennis; Shilts, Aaron
NetApp - A Client/Server Application Builder -- [+]
Hakimzadeh, Hossein; Hartman, Steve
Surfin' the Net: A Personal Finance Class Project --- [+]
Hallquist, Mary
Networking Option within the MIS Concentration --- [+]
Hawk, Stephen
Organizing for the University Web Site -- [+]
Hawk, Stephen

A Study of Techniques for Introducing Concurrency into List Processing Library Routines in Modern Concurrent Functional Programming Languages ------------------------------------ [+]

Heeringa, Brent; Lewandowski, Scott

Integrating Technology into a Computer Information Systems Classroom Instruction—What Works and What Doesn't -- [+]

Hettiarachchy, Jay

Virtual Lecture, Virtual Laboratory, or Virtual Lesson? -- [+]

Hill, Curt; Slator, Brian

Advanced Microcontroller Interfacing Laboratory --- [+]

Infanger, Bryan; Gammill, Robert; Tareski, Val

Portable Cluster Management (Panel) -- [+]

Johnson, Cj; Stenehjem, Stephen

A Case-Based Design for a Game Player --- [+]

Juell, Paul; Paulson, Patrick

Converting a Course's Presentations from Legacy Material to the Web ------------------------ [+]

Juell, Paul

Year Three of NDSU's Instructional Web Project -- [+]

Juell, Paul; Ross, James

A Course on Genetic Algorithms for Seniors and Graduate Students --------------------------- [+]

Julstrom, Bryant

Beyond a First Course in Windows Programming: Why, How, and What? --------------------- [+]

Kelsh, James; Lee, Roger

Time & Space Independent Learning: On-Line Course Offerings at a Small Liberal Arts College- A Status Report --- [+]

Konneman, Michael; Thomas, Lynda; Isaacs, Gerald

Bringing Streaming Media to Campus --- [+]

Korb, Aaron

Design Patterns in Java: *Sutherland* A Software Framework for Evolutionary Computation -- [+]

McPhee, Nicholas; Hopper, Nicholas; Reierson, Mitchell

Senior Research Project -- [+]

Mehta, Jean

Security and Flexibility: Computer Labs in a Small College ------------------------------------ [+]

Olson, Kevin

A Pilot Study: Offering a Web Course at a Small, Liberal Arts College ------------------------ [+]

Partridge, Jamie

Algorithm Animation using Java --- [+]

Patten, Dean

A Superdisplay Project --- [+]

Pavicic, Mark

Internet Research in the Humanities: Teaching Critical Thinking to First Year Students ------ [+]

Pederson, Molly; Sprunger, David

Smiley: A Web-based Remote Sensing Data Mining System ------------------------------------- [+]

Perrizo, William; Chen, Longjun; Amundson, Dennis; Li, Shuxue

Application of Active Networks in Distributed Systems -- [+]

Rahman, Faruq; Gammill, Robert

Surviving SPSS on the Desktop --- [+]

Roraff, Catherine

A Team Approach to Multimedia Design, Development, Testing and Evaluation: NDSU's New Center for Academic Information Technology -- [+]
Ross, James

Preparing Computer Science Graduates for the PC World --------------------------------------- [+]
Russo, John

Development of Daily Homework and Quiz Manager (DHQM) for Enhancing Teaching/Learning in Large Classes -- [+]
Schlecht, Nem; Mehta, Sudhir

Web and Database Integration for the Delivery of Dynamic Information (on a Budget) ------- [+]
Schlecht, Nem

From On-Line to On-Task: Communicative Technology in the First Year Classroom --------- [+]
Schnell, Carolyn

Planet Oit: a Virtual Environment and Educational Role-playing Game to Teach Geosciences --- [+]
Slator, Brian; Schwert, Donald; Saini-Eidukat, Bernhardt; McClean, Phil; Abel, Jon; Bauer, John; Gietzen, Brian; Green, Nathan; Jia, Yongxin; Kavli, Tammy; Koehntop, Lucas; Marthi, Bhaskar; Nagareddy, Vidyalatha; Olson, Acey; Peravali, Kishore; Turany, Daniel; Vender, Brad; Walsh, James

University of North Dakota Academic Affairs Web Initiative Pilot ----------------------------- [+]
Smart, Kathy; Pottenger, Steve; Wessman-Downey, Vicki; Olson, Bette

Impact of Technology on Quantitatively Oriented Business Classes --------------------------- [+]
Sounderpandian, Jayavel

Multi-facet Distance Education Services for Educational Leadership (Panel)-------------------- [+]
Stammen, Ronald; Nygren, Burton; Van Berkum, Dennis

Technology, Learning, and Equity Issues (Panel)-- [+]
Van Berkum, Dennis; Nygren, Burton; Sologuk, Sally

Using Hash Tables to Investigate Zipf's Law -- [+]
Weiss, John; Johnson, Roger

Five Exercises for Incorporating Moral and Ethical Issues into the Computer Science Curriculum --- [+]
Wickstrom, Milton

Global Information Society: A Multidisciplinary Course -- [+]
Wickstrom, Milton

Academic Planning: IT as a Strategic Asset in Higher Education

William Graves
President of the COLLEGIS Research Institute
Senior Vice President of COLLEGIS
member of the COLLEGIS Board of Directors.

Campus expenditures on information technology have increased dramatically and added pressure to spiraling institutional costs. To transform these costs into investments in its future, an institution must learn to deploy Internet technologies wisely to increase the learning productivity of its instructional programs. This session will demonstrate new online communication tools and online learning resources that are affordable, easy to use, scalable, and educationally effective in the hands of instructors who are dedicated to increasing their students' opportunities for learning, whether in a traditional classroom context or in a new anytime, anyplace instructional model. These demonstrations will be framed by a set of trends and issues which shape today's educational environment and which should inform an institution's planning, budgeting, and faculty development programs as they relate to support for information technology.

Distance Education – Something Old, Something New, Something Borrowed, Something Blue

Dr. Chere Campbell Gibson
Continuing Education and Vocational Education
University of Wisconsin-Madison
Madison, WI

The words are recognizable to many as words heard by a bride prior to a wedding in modern Christendom. The title is not meant to be exclusionary, but to put a metaphor out on the table for consideration – a metaphor that brings to mind change, merger partnership, give and take, building on the best of those united, etc. The "something old" is distance education while the "something new" is represented by some of the newer technologies we've added to the mix. The main focus of the presentation addresses "something borrowed" from traditional teaching venues, that being a model of teaching and learning, and the need to engage in a paradigm shift in the roles and responsibilities of teachers and learners as we engage in distance education. If change is not forthcoming, we'll discover "something blue" – learners who, through the advent of modern technologies, have gained access to information yet are not able to be successful in accomplishing their learning goals. In this competitive educational market place some marriages of institutions and learners will survive. Many will not. Free counseling provided.

Security Policies: What Campus Managers Need to Know

Melanie Abbas
Systems Administrator – College of Natural Sciences
Part-Time Computer Science Graduate Student
University of Northern Iowa
abbas@uni.edu

One of the main concerns of information technology management today is security. IT Managers need to be aware of the security of their internal networks. A security policy is critical to any organization that uses any significant amount of information technology. There is a straightforward process to creating a security policy.

A security policy needs to contain a minimum of five elements [Bruce and Dempsey 1997]. Those elements are

1. Policy Statement - the specific policy the organization wants to address
2. Purpose - explains why the policy is being addressed
3. Scope - details how far the policy extends throughout the organization
4. Compliance with policy - discusses what is necessary to comply with the policy
5. Consequences of non-compliance - lists the formal sanctions undertaken when there is a breach in the policy

Security Policies: What Campus Managers Need to Know

Melanie Abbas
Computer Support Specialist – College of Natural Sciences
Part-Time Computer Science Graduate Student
University of Northern Iowa
abbas@uni.edu

Introduction

One of the main concerns of information technology management today is security. Just how secure is the internal network, intranet, or the Internet connection of the organization? What kinds of activities should employees be able to do from their office personal computers? How should employee activities be monitored? Who is really responsible for overseeing these concerns?

These are all issues that a security policy can address for an organization. That is why the development of such policies is very critical to technology intensive operations. The security policy must be carefully planned, broadly advertised, widely supported, and strongly enforced in order for the policy to make a difference in an organization. There are many resources available to help management work with top executives to develop an organization wide policy that is right for them.

While these areas are becoming well-documented for use within business organizations, in the area of education, the idea of policy development is still in its infancy. In building the virtual campus, what kinds of activities should be encouraged, discouraged, or completely restricted is a very difficult decision. What kind of security risks are a university or college willing to allow in order to promote distance education? How much flexibility do students, faculty, and staff need to do their work?

Security policies are under evaluation on our campus as well as many around the country. Within the University of Northern Iowa campus, this expansion of services included local Internet service providers and key network personnel on campus. The local service providers are unique in that they allow direct network connections to households or businesses within the campus community. They too have been working through legal documents to ensure that they are doing what is needed to provide service and protect their systems.

Our campus started a pilot project to extend services to dorm rooms this fall, and has completed one dorm. Plans are being made to expand this to all dorms on campus. This expansion brings up policy questions for any network administrator on campus. What kind of network traffic will it cause? What about licensing issues with software intended for campus labs being used by dorm occupants? Will anyone patrol the dorms to find out how much software is being copied onto student's personal computers? How about highly-skilled students who can sit in their dorms all night and try to hack into campus systems? Campus network managers face an ever-changing environment and security policies can help them address these issues in a consistent manner.

Why is Security Such a Hot Topic

With the explosion of the Internet, the amount of information flowing on networks around the world is astounding. Any computer user, even those with little technical skills, can setup their own computer as a server. Every server is a potential entry point to the system and, thus, it is very important to know what kind of access is allowed on any given connection to the Internet. Access control is used to limit actions available to a user on a given system, server, segment, or connection based on user identity.

On the WWW, the identity of the user is very difficult to determine. Many organizations are putting more information on-line with Web servers. Even the Department of Justice has found this can be more difficult than expected. On August 16, 1996, hackers defaced the Department's WWW pages and caused them to rethink their security configurations [Cox 1997].

However, security measures only address known breaches, not unknown or as yet undiscovered break-ins. Therefore, security managers must keep current on new breaches and guard for possible new break-in paths. There are mailing lists and other Internet groups dedicated to listing new breaches, such as the one listed about General Electric in 1994[Harp, Greg]. Some say it may be the largest security incident ever admitted by a

major corporation since most such incidents are supressed. Apparantly, an intruder from the Internet broke into their internal network browsing business information on NBC and capturing some passwords. In order to guard against such breaches, an organization needs to support a security policy throughout the organization's network connections.

Who is Responsible for the Security Policy

The question that follows is how the security policy is to be addressed or created and by whom. The security of information and resources for an organization is the responsibility of the Information Technology (IT) management. However, in order for the IT department to enforce security, it is necessary to have a set of guidelines or standards that the organization is proposing to uphold. This set of standards needs to be written into a formal security policy for the organization. It is important to involve people from all levels of the organization during the creation process to ensure that everyone is fairly represented[Curran 1997].

Often, a site has developed a computer security policy with involvement of only the IT administration addressing only limited concerns. There are many other methods for developing security policies and organizations that are available to perform in depth consulting. These companies are highly advertised on the WWW for the specialty areas. For example, NJH Security Consulting Inc.[NJH Security Consulting] provides network security consulting, VaultBase [VaultBase] provides data security consulting, Qualtec Inc.[Qualtec] handles all kinds of physical security devices.

Although these consulting and systems security integrators are a great resource, it is important to remember that nobody can create an intruder free network, no matter what they promote [Cox 1997]. The information from the consulting should be used to assist internal employees in creating a security policy. For network administrators, having a firm written policy is a blessing, since it gives them the freedom to use what measures they deem necessary to enforce the policy or allow breaches dictated by the policy.

What Does a Security Policy Contain

A security policy needs to contain a minimum of five elements [Bruce and Dempsey 1997]. Those elements

are

1. Policy Statement - the specific policy the organization wants to address
2. Purpose - explains why the policy is being addressed
3. Scope - details how far the policy extends throughout the organization
4. Compliance with policy - discusses what is necessary to comply with the policy
5. Consequences of non-compliance - lists the formal sanctions undertaken when there is a breach in the policy

It takes several of these individual policies to be incorporated into a master security policy for an organization. The individual policy statements are often the easiest part of the security policy. The purpose often follows from the statements. However, the scope, compliance and consequences are the details of the policy that are critical in order for the policy to be effective. Table 1 is an example of a basic security policy:

Network Login Policy

Company X requires that all users have one and only one network login to the central network server that is not shareable by more than one individual or by an individual on more than one workstation at a time. (This is the policy statement)

Purpose

Many of the software packages that are in use in Company X, require individual settings that can only be used by one individual at a time. Company X also performs auditing on all activities on the main network and relates them to an individual login. Thus, each individual is responsible for every action taken by their network login. If a user needs to work in a group with multiple users, the network administrator needs to create a group and place all necessary users in the group to avoid creating new group logins.

Innocent violations of the policy may include:

- Sharing login names and passwords with family members.
- Allowing subordinates to login as an individual to check e-mail
- Leaving a workstation logged in and unattended for another employee to access the network
- Creating shortcuts or caching passwords such that anyone could access the network without entering passwords
- Violating any of the good password policies (in another policy)

Scope

All employees are subject to this policy, except IT personnel who are allowed multiple simultaneous logins when doing installation and configurations.

Compliance with Policy

Ignorance of this policy can result in non-compliance and is punishable as noted in the Consequences section. To assure compliance with network logins the user must

- Ask the network administrator to create one and only one account for his/her access to the network
- Keep passwords private and follow good password policies
- Always logout of a station or leave it in a locked state before leaving it unattended
- Never cache passwords
- Do not allow others to perform any activities on the network while the user is logged in

Consequences

Practices that create a state of non-compliance by a user are punishable by removing all access to the network from an individual. The user may regain network access depending on the circumstances and discretion of the network administrator.

Table 1: Sample Network Login Policy

This policy is addressing the issue of network login accounts on Company X's network. The purpose is clearly stated. A list of innocent violations is included to clear up confusion when people believe these acts are not offenses. The scope addresses the issue of who should be included in the policy and leaves exception for IT personnel who may have special needs. The compliance section lists exactly what needs to be done by a user in order to fulfill the requirements of the policy. Finally, the last section puts in writing the consequences of breaking the policy. The consequences are very general to allow the punishment to be left up to the enforcer of the policy.

Since heated security discussions are often very difficult to settle, many organizations never actually create a written computer security policy, but simply believe that the users know what they can or can't do. This is a dangerous assumption. Therefore, the IT management needs to know the best method to create a written policy which they can defend themselves in a case of loss due to a security breach.

What Does a Security Policy Need to Protect

The first question asked to create a security policy is what needs to be protected? It could be data, resources, privacy, communication lines, or all of these. The type of information being protected determines the actual policy statement. Some of the questions that are commonly used to create a security policy include

- Who is allowed accounts in Company X?
- Are guest accounts allowed?
- How do employees gain access to services from the main network in Company X?
- Is remote dial-in access allowed?
- What kind of virus protection is required?
- Are remote accesses necessary to the main network of Company X?
- What information is confidential?
- Are users allowed to transmit internal information via e-mail?

The first questions address the issue of access to the internal IT resources. Different organizations have different expectations in regards to access. The resources may be open to the public or highly classified; thus classifications of access may be necessary. Dial-in access is another main concern due to the ability of hackers to find modem banks to attack in hopes of finding a connection.

Other concerns, such as virus protection are more associated with the data that the organization maintains. If even one workstation without virus protection with a virus in memory accesses data, all the data can become corrupt. This can cause major business failures due to loss of records. Additional destruction can occur if the information is confidential and an internal user is allowed to e-mail the information outside the organization.

After these questions are answered, a list of policy statements can be created. It is then a difficult task of stating the purpose, scope, compliance, and consequences of the policy. This is often a management decision that is best made by a committee including top management executives, IT personnel, and end users. If the Security Policy can be incorporated into the Strategic Plans of an organization, it will be more respected and followed. Often these plans increase the budgeting necessary to enforce the security policies[Chapman and Swicky 1995].

Enforcing Security Policies

Having a policy stating what is acceptable and not acceptable is good, but not effective unless measures are put in place to enforce the policy. In the consequence portion of the policy, there must also be a list stating the repercussions of breaking a policy. It is best to leave these as general consequences since the degree of the breach may determine the actual punishment. The task of enforcing the policy is usually done by IT personnel who were not directly responsible for the creation of the policy. This avoids any conflicts of interest.

Enforcing is a time-consuming task for many policies, but others may be addressed by software and hardware security configurations. However, if the policy is important to an organization, they must learn to budget accordingly for staff and/or hardware/software solutions. One of the most highly advertised retail solutions provider in this area is On Technology [11]. They provide off the shelf hardware and software solutions, one of these products is a firewall. A firewall is "a component or set of components that restricts access between a protected network and the Internet, or between other sets of networks" [Chapman and Swicky 1995].

A firewall takes some configuration and maintenance, but is a great tool for not only restricting actions, but also monitoring actions of the users on the network. It is understood that all the actions taken on the network are for the best interests of the company, therefore the firewall manager is allowed to track system activity. The firewall manager may or may not be the person who enforces the policies.

Another issue that results when enforcing a policy with a firewall is the legal ramifications it involves. If a company uses a firewall to limit all types of communication, if any communication breaks the firewall, the company is responsible for the actions taken. However, if the firewall is not used, and a written policy is created, the individual breaking the policy is legally liable for their actions.

This is a major concern of the IT personnel, since a firewall can never be completely foolproof. One answer to this situation is to create a written policy that individual users are responsible for their actions regardless of the security mechanisms that are in place. "It should be clearly stated that breaking into accounts or bypassing security is not permitted." [Curran 1997]

Publicizing Security Policies

An organization's Security Policy needs to be formalized into the standard policies and procedures of the organization. If a policy manual exists, it should be included in the employee manual and distributed to new or future employees along with explanation as any other policy. If the organization has an intranet, that is an ideal place to keep a current copy of the policy and notify employees when a change has been made. A formal hard copy of all security policies should be kept by the IT personnel that enforce the policies on a regular basis.

Accepting the Security Policy

There will always be someone in the organization that feels that a particular part of the security policy is too restrictive. There are also employees that are trustworthy enough to have more privileges than the average user addressed in the policy. That is why, in some cases, it is necessary for portions of the security policy to list the exceptions for an individual policy. For example, an organization has a security policy forbidding any remote dial-in access to a given network. However, there are several employees that perform much of their work on the road and need access to the network to keep efficient communication with other employees. In this case, it may be that the policy can state that exceptions to the dial-in access are allowed, if a given form is filled out and approved by a security enforcer and/or an IT department manager. This policy is still a good policy, since all violations require specific documentation that keeps them in compliance with the policy.

Summary

IT Managers need to be aware of the security of their internal networks. A security policy is critical to any organization that uses any significant amount of information technology. There is a straightforward process to creating a security policy. The key to creating a policy that both works and is accepted by the majority is to involve individuals from all groups in the organization. There are many companies on the Internet that will assist a company in evaluating their IT security and creating a policy, if the company cannot do it on their own.

Case Studies

Boston College Dorms

Boston College[Boston College] implemented full network data access along with other services to all their dorms during 1995. They planned on migrating the technology for students living off campus using cable modem technology similar to that which was installed in the dorms.

The program appeared to be a success. Many other universities, including the University of Northern Iowa, are planning on similar implementations. The issues are so complex that a central resource area has been created at Stanford to address security policy issues called Resnet[10]. On Resnet the universities post similar policy questions and answers along with a copy of their security documents for others to use as a model. It also contains links to other security resources, such as those listed below.

University of Northern Iowa Campus

The University of Northern Iowa has successfully brought 2 dorms onto the campus network. By fall of 1998, they plan on having ethernet access from all the dorms to centeral resources on the campus. The goal is to provide a working environment in a dorm room that will duplicate that of a campus computer lab. This goal is to extend to all faculty and staff on campus such that access from any computer on campus will allow an employee to gain access to all resources they require to do their job.

The fight is just beginning for campus IT managers who think they have control over who is on their network. At this time, most central college servers are filtered from the dorm network to avoid security concerns. However, this approach is going to have to change if the total campus solution is to succeed. Now more than ever, network security is going to be a vital part of continuing education for campus IT managers.

Relevant WWW Security Resources

- Department of Defense Trusted Computer System Evaluation Criteria (URL http://ftp.sterling.com/tsig/references/orange-book/orange-book)

- Innovative Security Products (URL http://www.qni.com/~lidlock/ispfyi.htm)
- Security Breaches (URL http://www.ncc.co.uk/secbreac/page1.html)
- Security Policy Handbook for Managers (URL http://www.versalink.com/Sanda/Pubs/cs/94-01.txt)
- Texas A & M Security Policy (URL http://www.tamu.edu/reports/appendix-m.html)
- Working Security Policy Handbook (URL http://www.ix.de/doc/cert/ieft/ssphwg/ssph-draft-28nov.txt)

References

[Bruce and Dempsey 1997] Glen Bruce and Rob Dempsey, *Security in Distributed Computing:Did You Lock the Door?*, Hewlett-Packard Professional Books, 1997.

[Cox 1997] John Cox, *A Web Lesson Learned*, NetworkWorld, Vol. 14 No. 30, July 28, 1997, p. 1&68 http://www.nwfusion.com.

[Curran 1997] Curran, Charles. "Establishing Official Site Policy on Computer Security." (URL http://users.ox.ac.uk/~charles/RFCs/1244/rfc1224-2.html)

[Chapman and Swicky 1995] D. Brent Chapman and Elizabeth D. Swicky, O'Reilly & Associates, Inc. *Building Internet Firewalls*, 1995

[Harp, Greg] "FireWall Mailing List: Security Breach at GE?" Online posting, <http://www.netsys.com/firewalls/firewalls-9411/0524.html>, 1994

[NJH Security Consulting] "NJH Security Consulting Main Page" <http://www.njh.com/>,1997

[VaultBase] "Database Security – by VaultBase <http://www.vaultbase.com/page/SecurityPage.html>,1998

[Qualtec]. "Qualtec Main Page"<http://www.pc-security.com>, 1998

[Boston College] "Cable Modem Technology at Boston College". <http://www.bc.edu/agora/cablemodems.html>, 1995

[10] Resnet - Residential Network Issues. (URL http://rescomp.stanford.edu/others.html)
[11] On Technology. (URL http://www.on.com)

Virtual Dragon Days:
Web-based Registration and Orientation

Jeanne Alm
Computer Center
Moorhead State University
Moorhead, Minnesota
alm@mhd1.moorhead.msus.edu

Les Bakke
Computer Center
Moorhead State University
Moorhead, Minnesota
bakke@mhd1.moorhead.msus.edu

Stacy Grim
Computer Center
Moorhead State University
Moorhead, Minnesota
grim@mhd1.moorhead.msus.edu

Virtual Dragon Days utilizes the Internet to provide an alternative orientation and registration method for institutions of higher education. Moorhead State plans to use the transfer student cohort to pilot the system. Section one defines the components of *Virtual Dragon Days*. The proposed elements include an on-line college catalog; on-line advising; an on-line campus tour; on-line financial aid application and status reporting; and web registration. Section two discusses the on-line catalog and campus tour. Section three presents advising elements that must be designed to effectively replace conventional face-to-face contacts. Section four reviews financial aid requirements. Finally, section five presents the capstone of the *Virtual Dragon Days* project, the web registration system. Initially this project is expected to provide transfer students with an orientation and registration experience that is not currently available to them. In the future, the system can be expanded to include other segments of the incoming student population.

Virtual Dragon Days:
Web-based Registration and Orientation

Jeanne Alm
Computer Center
Moorhead State University
Moorhead, Minnesota
alm@mhd1.moorhead.msus.edu

Les Bakke
Computer Center
Moorhead State University
Moorhead, Minnesota
bakke@mhd1.moorhead.msus.edu

Stacy Grim
Computer Center
Moorhead State University
Moorhead, Minnesota
grim@mhd1.moorhead.msus.edu

Introduction

Dragon Days are those days scheduled for new students to come to the Moorhead State University campus, meet with advisors, register for classes, check on financial aid and tour the campus. *Virtual Dragon Days* provide a similar experience via the Internet.

Virtual Dragon Days will use the Internet to provide an alternative method of orientation and registration for incoming college students. The project structure combines the existing web components—on-line college catalog, campus tour, web registration, and financial aid application forms—with new elements. The additions will add financial aid status reporting and an on-line advising system that provides transfer credit evaluation and e-mail contact with academic advisors and admissions counselors. In this paper we hope to provide other institutions searching for innovative means of reaching new and prospective students with a summary of our project and advice on things that worked well and, perhaps, those that did not.

Section one defines *Virtual Dragon Days* and introduces its components. Existing elements—on-line college catalog and campus tour; on-line financial aid applications; and web registration—must be augmented by an on-line advising system and financial aid status reporting. Section two discusses the on-line catalog and campus tour. Section three addresses the need to effectively replace conventional face-to-face contacts with an on-line advising system. An existing database of many regional institutions' courses allows transfer credits to be evaluated from the student's self-reported coursework. Additional admissions counseling and academic advising will be accomplished with regular e-mail contacts. Section four reviews the system's financial aid requirements. Financial aid applications are currently accepted on the web; however, a financial aid status report, incorporating a projected budget with any financial aid awards, must still be developed. Section five presents Web Registration. Introduced in October 1997 for Spring 1998 pre-registration, Web Registration provides 24-hour access for students to enroll in classes from any computer connected to the Internet. It includes student class schedule and grade lookups as well as an on-line version of the university term schedule with current class availability. Throughout this document, we reference the screen images found in Appendix A. The conclusion contains a brief summary of *Virtual Dragon Days* and a glimpse at our future plans for the project.

Moorhead State University will pilot *Virtual Dragon Days*, expected to be available for the 1998-1999 academic year, with the transfer student cohort. Initially the project is expected to provide our transfer students with an orientation and registration experience that is not currently available to them. In the future, the system can be expanded to include other segments of the incoming student population.

1. Components of the *Virtual Dragon Days* System

At Moorhead State University, *Dragon Days* are those days scheduled for new students to come to campus, meet with advisors, register for classes, check on financial aid and tour the campus. *Virtual Dragon Days* is designed to provide a similar experience via the Internet. The following sections explain the components of *Virtual Dragon Days*. Section two provides information on the on-line catalog and campus tour. This information is important not only to currently enrolled students, but also to prospective students attempting to make the correct college choice. Section three provides information on enhancements to the traditional methods of admissions counseling and academic advising. Both are brought into the electronic age. Section four describes how financial aid application and counseling will be a vital part of *Virtual Dragon Days*. Section five is the most critical component of *Virtual Dragon Days*. Web registration, checking for open sections, viewing grades and retrieving schedules will complete the enrollment cycle.

2. The On-line Catalog and Campus Tour

This section discusses the on-line copy of the college catalog, which is essential to the project, and the on-line tour of campus facilities and student services, which offers Internet visitors an introduction to the university.

On-line College Catalog

An on-line copy of the college catalog serves many purposes; therefore, it must be scheduled at the beginning of the project. An electronic catalog must be organized in a way that allows prospective and admitted students to easily access its information. A university must decide if the on-line catalog will be an *electronic copy* of the printed document supplemented with additional information, or if the on-line version will attempt to provide the most current information. In our case, the on-line catalog was initially updated as changes occurred; however, there was significant confusion over which document was *correct*, the printed *Bulletin* or the web information. The current policy at MSU is to create an electronic copy of the document at the time of publication. Any changes are presented as *Addendum* or *Errata* links to corrections authorized by the Records Office.

The on-line Moorhead State University *Bulletin* begins with an electronic copy of the *Table of Contents* (see Fig.1 in Appendix A), with each entry linking to its respective pages. Cross-referenced items are actively linked to each other allowing readers to easily track any topic. The on-line *Bulletin* contains some general information about Moorhead State, university policies and procedures, curricula and course information, and a listing of administration and faculty. Both undergraduate and graduate*Bulletins* are on-line; however, we will limit our discussion to the contents of the undergraduate catalog.

In addition to a brief history of the university, the electronic *Bulletin* describes MSU campus facilities and services, provides information on student policies, specifies admission and financial procedures, and lists academic programs. The *Curricula* section provides a complete description of requirements for each major and minor program within its affiliated department. The*Courses* section provides an alphabetical list of university courses by subject name and number. This section also allows prospective students to search for courses by subject name and/or subject number, or by keywords in the course title as shown in Fig. 2 of Appendix A. Course information includes subject name and number, course description, number of credits, pre-requisite or co-requisite courses, and General Ed designation. In summary, the on-line catalog offers those with Internet access immediate access to the current MSU *Bulletin*.

On-line Campus Tour

An on-line campus tour should highlight campus facilities and introduce the services available to students. Utilizing a combination of graphics and text, a campus tour provides prospective students with a *feel* for the university prior to their first visit. It is important to make this first impression of your campus a positive experience.

The Moorhead State University on-line campus tour begins with a color-coded map of the campus showing the location of classroom buildings, dormitories, and parking lots. The pages associated with the classroom buildings contain links to the departments located within each building as illustrated in Fig. 3 of Appendix A. Department pages, in turn, link to the *Bulletin* pages containing the programs and curricula associated with each department. Department pages also link to the department's home page, if one exists.

Departments, faculty, and student organizations are encouraged to maintain their own homepages which supplement the "official" catalog information. Many departments have taken advantage of this opportunity to exhibit their unique offerings. A number of faculty use their home pages to distribute class materials—syllabi and assignments—electronically.

3. On-line Admissions Counseling and Academic Advising

Several traditional methods of interacting with students will be enhanced with on-line admissions processing. Prospective students have the opportunity to browse the Internet for college choices. By using the On-line Catalog and Campus Tour, prospective students may visit our campus from throughout the country. Using e-mail these same students will ask questions, receive answers, solicit opinions from others, and make their college choice. Admissions counselors, academic advisors and faculty will be required to become more active providers of electronic admissions information.

Marketing of the university will include scheduled e-mail messages along with the traditional recruitment plans. Currently MSU uses targeted mailings, mass mailings and telemarketing to recruit students. E-mail messages will add another planned contact with the prospective student. Because e-mail is immediate, messages related to time sensitive events becomes easy. For instance, if a prospective student is playing on a basketball team that just won the regional tournament, a congratulatory message could go out the same day.

This section presents the equally important advising element that must be designed to effectively enhance conventional face-to-face contacts. Transfer students need to know how courses already taken will *fit* in their program at the new institution. MSU has over 11,000 courses from other colleges on-line. A recent project with the faculty and admissions resulted in the analysis of those courses. An equivalency table shows what local course is the same as the transfer course as shown in Fig. 4 of Appendix A. Students at two-year colleges have the opportunity to make their course choices based on how the course will transfer.

Once a student has been accepted, traditional advising will be supplemented by providing e-mail answers to transfer student questions. Selected faculty will be given the opportunity to become the leaders in on-line advising.

4. On-line Financial Aid Application and Counseling

Financial Aid requirements for the on-line system include electronic forms to apply for financial aid and scholarships, as well as an electronic means of performing different aspects of financial aid counseling.

MSU currently has a financial aid web page that contains a link to the Free Application for Federal Student Aid (FAFSA) site, where students can electronically complete their financial aid application. University applications for upperclass and freshman scholarships are available on-line. Students may print the forms, complete and sign them, and mail them to the Financial Aid office. Students will soon be able to view their financial aid status as shown in Fig. 5 of Appendix A.

The financial aid status report will list a student budget for the specified academic year: the estimated costs less the student and parent contributions to equal the total estimated financial needs. Next it lists the type and amount of each grant and/or loan that the student has been awarded. Finally, students will be able to complete entrance and exit loans on-line.

5. Web Registration

The ability to perform registration tasks—add and drop courses, search for open course sections, view a class schedule, check for registration holds, and view term grades—is an essential part of the system. This component is not only helpful to incoming students, but is a strong attraction to students searching for a university. With registration tasks on-line students avoid the traditionally long registration lines. A student may now register from across campus, across town, or across the country at any time of the day or night.

The MSU Web Registration system requires three items of information to login into the system: an eight-digit student Dragon Id, a six-digit PIN, and an Advisor Access Code. Students are able to update their PIN number at any time. An Advisor Access Code is required for any student who is required to see an advisor. Advisor Access Codes are changed every semester.

Using the above information, a student will be able to retrieve the current web registration menu options: view class schedule, check grades, check registration time (window), check for student holds, change PIN, check for open classes, and register for classes.

The first time students use the registration process they *must* change their PIN from a preset birthdate code to a familiar six-digit number. Next they will check to see if there are any holds on their record. If they have holds, office phone numbers are listed so they can arrange to remove the hold and register. Students can check the registration window so they know what time they can begin to register. Once their registration window is open, they can select the *Register for Classes* menu option where they can easily add or drop up to six courses by selecting the desired option and entering the Course ID, as illustrated in Fig. 6 of Appendix A. Students must re-enter their PIN for each registration change request. Once the correct PIN is re-entered, the registration requests are sent to the database. The students' current class schedules are displayed reflecting any changes. Error and warning messages will appear if the request can not be completed.

Students also have the option of searching for open classes. At the registration screen, they can enter the course subject and/or subject number and/or General Ed code for which they wish to search. A list of open courses will be displayed. The student may then just click on any course in the list to register. This feature is very helpful when planning a schedule or when class sections begin to fill.

Faculty will access a different application to perform registration overrides as shown in Fig. 7 and Fig 8. Using the Internet, authorized faculty can grant permission for students to enroll in closed classes or classes with special restrictions, such as minimum GPA, specified major or specified program.

Conclusion

To conclude this paper, we will share some plans for future development of the *Virtual Dragon Days* project and then summarize our work to date.

Future Plans

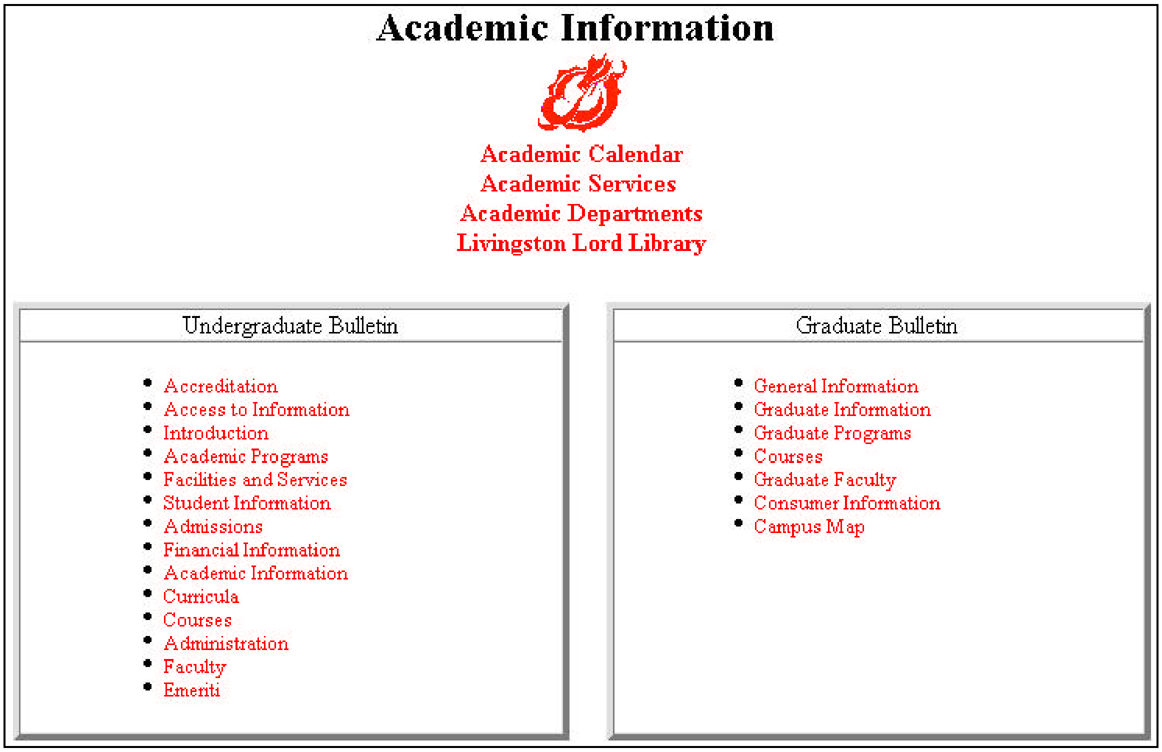
Future development of the *Virtual Dragon Days* project will branch in two directions, expanding the target audience and expanding available features of the system. Initially this project is expected to provide transfer students with an orientation experience that is not currently available to them through any means. In the future, the system can be expanded to offer an alternative method of orientation and registration for all segments of the incoming student population.

In the future, features of *Virtual Dragon Days* may be expanded to include such areas as housing and billing. Housing inquiries and applications could easily be processed via the Internet. Another possibility is an application that would allow students to pay their tuition and housing bills with a secure connection on the Internet.

Summary

The *Virtual Dragon Days* project was initiated to provide an alternative orientation and registration method for incoming students via the Internet. The elements of the system include an on-line college catalog; on-line advising; an on-line campus tour; on-line financial aid applications and status reporting; and web registration. Section two reviewed the existing on-line catalog and campus tour and discussed how they contribute to the project. Section three explained the admission counseling and advising tools that must be incorporated to effectively replace conventional face-to-face contacts. Section four reviewed financial aid requirements of the system. Finally, section five presented the web registration system, an integral part of the *Virtual Dragon Days* project.

Appendix A



Academic Information

Academic Calendar
Academic Services
Academic Departments
Livingston Lord Library

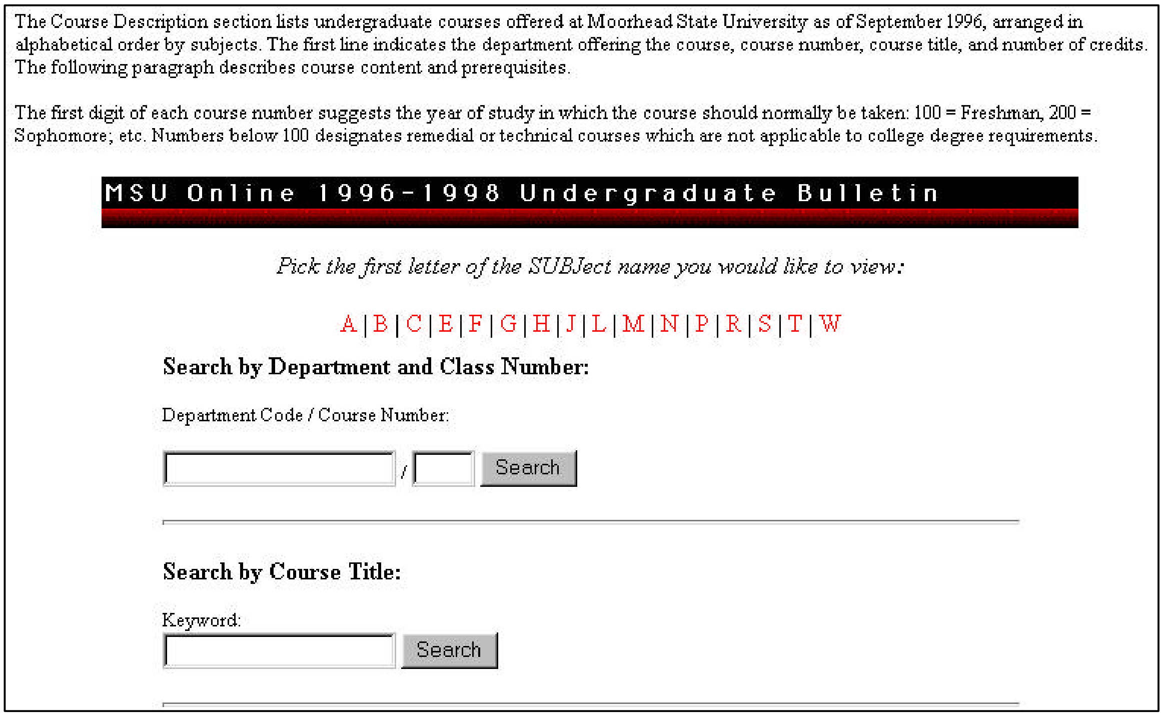
Undergraduate Bulletin

- Accreditation
- Access to Information
- Introduction
- Academic Programs
- Facilities and Services
- Student Information
- Admissions
- Financial Information
- Academic Information
- Curricula
- Courses
- Administration
- Faculty
- Emeriti

Graduate Bulletin

- General Information
- Graduate Information
- Graduate Programs
- Courses
- Graduate Faculty
- Consumer Information
- Campus Map

Figure 1: Table of Contents from On-line Bulletin



The Course Description section lists undergraduate courses offered at Moorhead State University as of September 1996, arranged in alphabetical order by subjects. The first line indicates the department offering the course, course number, course title, and number of credits. The following paragraph describes course content and prerequisites.

The first digit of each course number suggests the year of study in which the course should normally be taken: 100 = Freshman, 200 = Sophomore; etc. Numbers below 100 designates remedial or technical courses which are not applicable to college degree requirements.

MSU Online 1996-1998 Undergraduate Bulletin

Pick the first letter of the SUBJect name you would like to view:

A | B | C | E | F | G | H | J | L | M | N | P | R | S | T | W

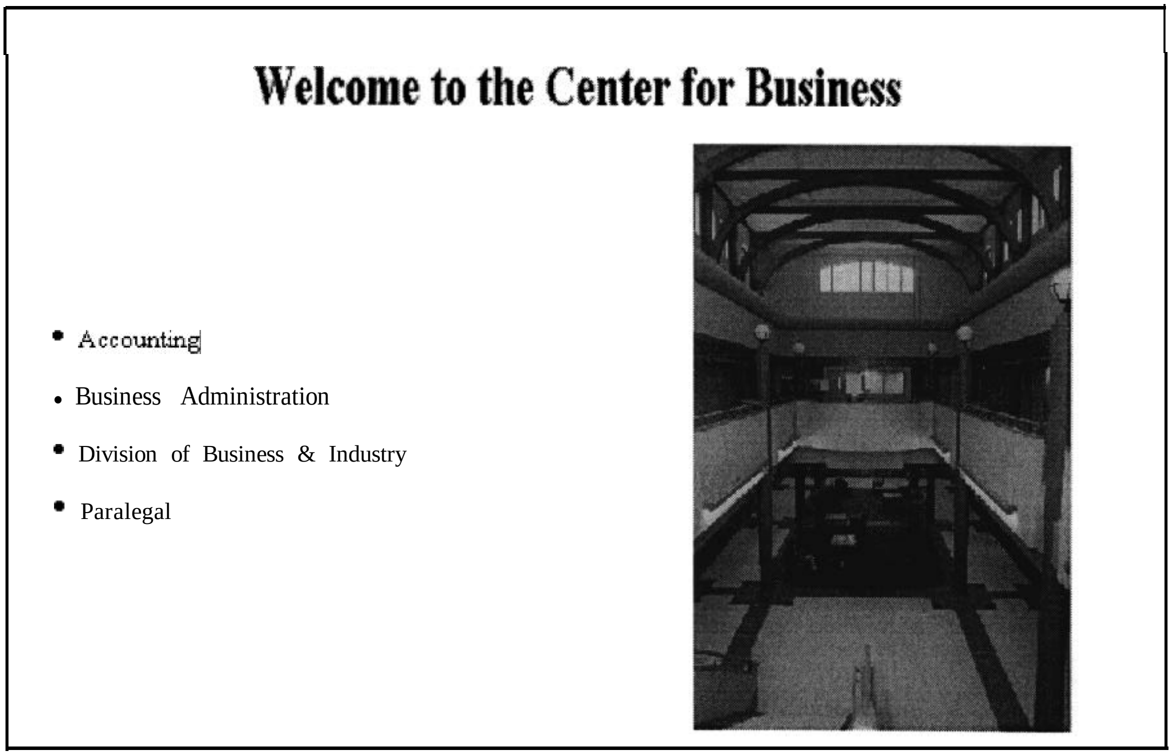
Search by Department and Class Number:

Department Code / Course Number:

Search by Course Title:

Keyword:

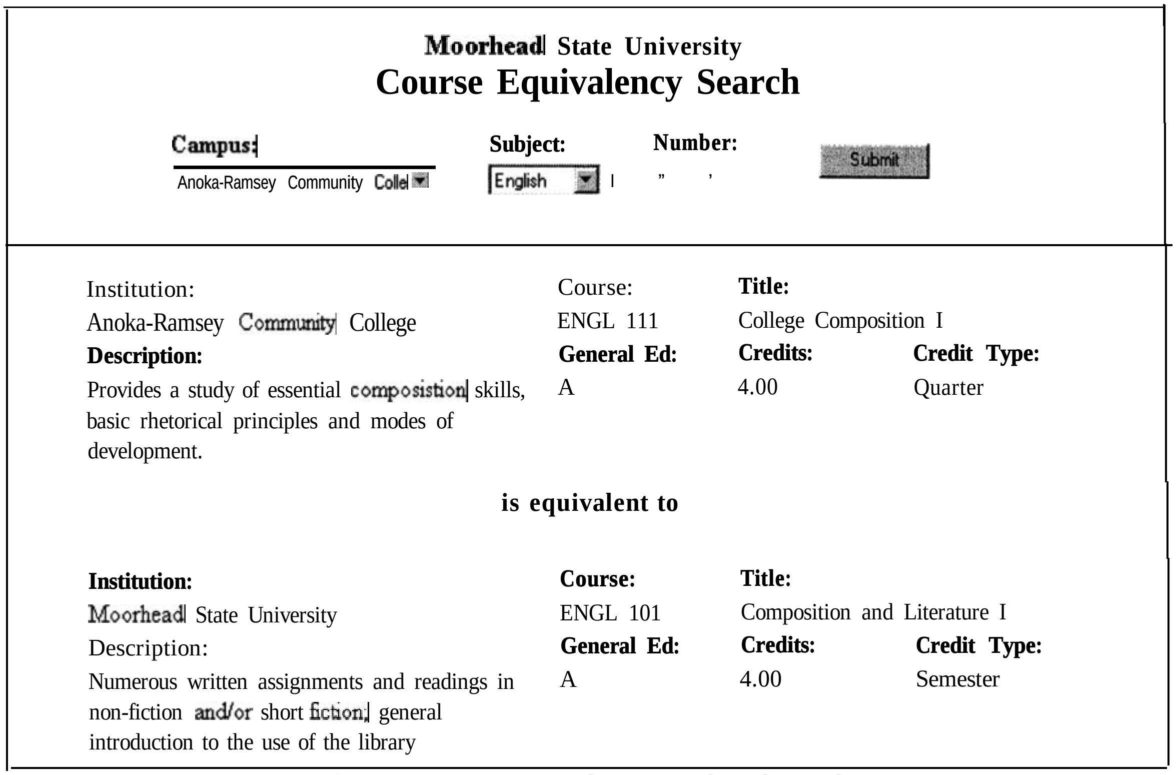
Figure 2: Course Search in On-line Catalog



Welcome to the Center for Business

- Accounting
- Business Administration
- Division of Business & Industry
- Paralegal

Figure 3: Typical Building Page on the Campus Tour



Moorhead State University

Course Equivalency Search

Campus:	**Subject:**	**Number:**	
Anoka-Ramsey Community Colle	English	" '	Submit

Institution:
Anoka-Ramsey Community College

Description:
Provides a study of essential composistion skills, basic rhetorical principles and modes of development.

Course:	**Title:**	
ENGL 111	College Composition I	
General Ed:	**Credits:**	**Credit Type:**
A	4.00	Quarter

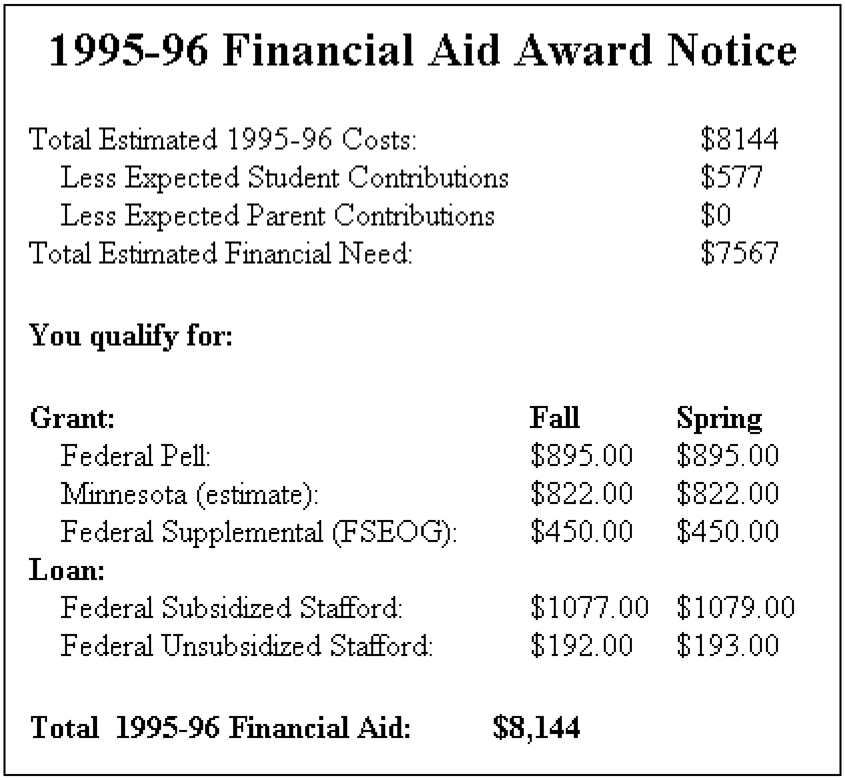
is equivalent to

Institution:
Moorhead State University

Description:
Numerous written assignments and readings in non-fiction and/or short fiction; general introduction to the use of the library

Course:	**Title:**	
ENGL 101	Composition and Literature I	
General Ed:	**Credits:**	**Credit Type:**
A	4.00	Semester

Figure 4: Course Equivalency Search with Results



1995-96 Financial Aid Award Notice

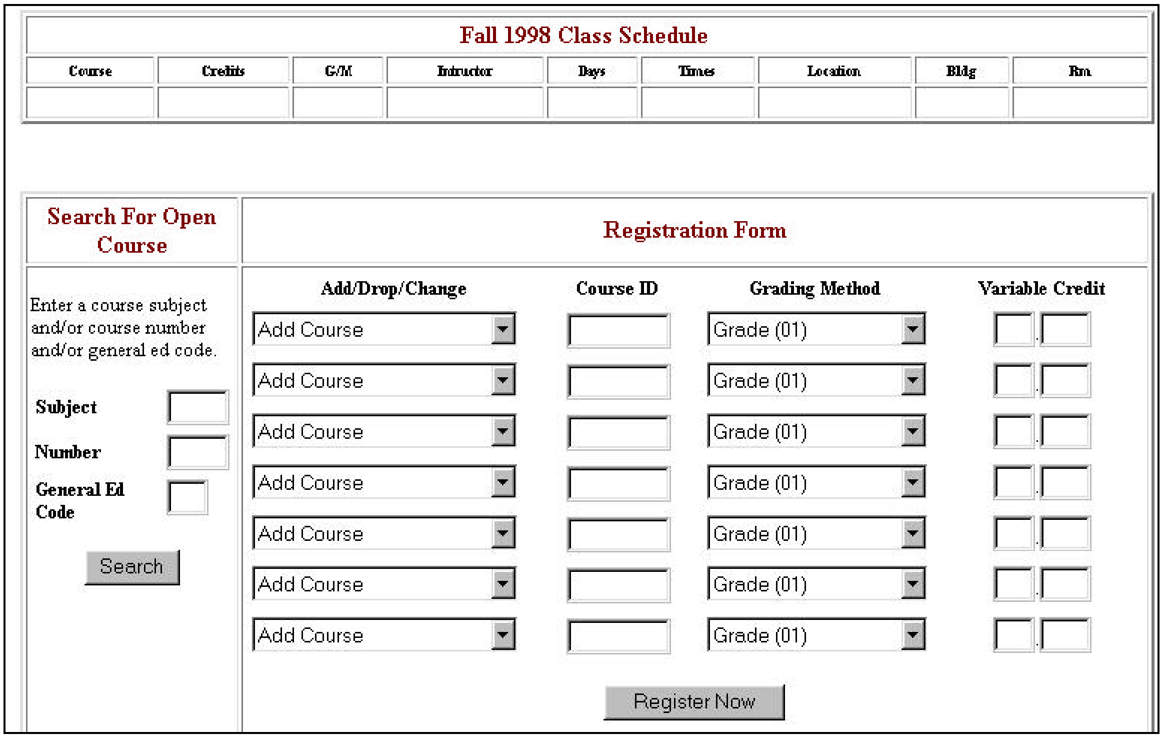
Total Estimated 1995-96 Costs:	$8144
Less Expected Student Contributions	$577
Less Expected Parent Contributions	$0
Total Estimated Financial Need:	$7567

You qualify for:

Grant:	**Fall**	**Spring**
Federal Pell:	$895.00	$895.00
Minnesota (estimate):	$822.00	$822.00
Federal Supplemental (FSEOG):	$450.00	$450.00
Loan:		
Federal Subsidized Stafford:	$1077.00	$1079.00
Federal Unsubsidized Stafford:	$192.00	$193.00

Total 1995-96 Financial Aid: $8,144

Figure 5: Typical On-line Financial Aid Award Notice



Fall 1998 Class Schedule

Course	Credits	G/M	Intructor	Days	Times	Location	Bldg	Rm

Search For Open Course

Enter a course subject and/or course number and/or general ed code.

Subject

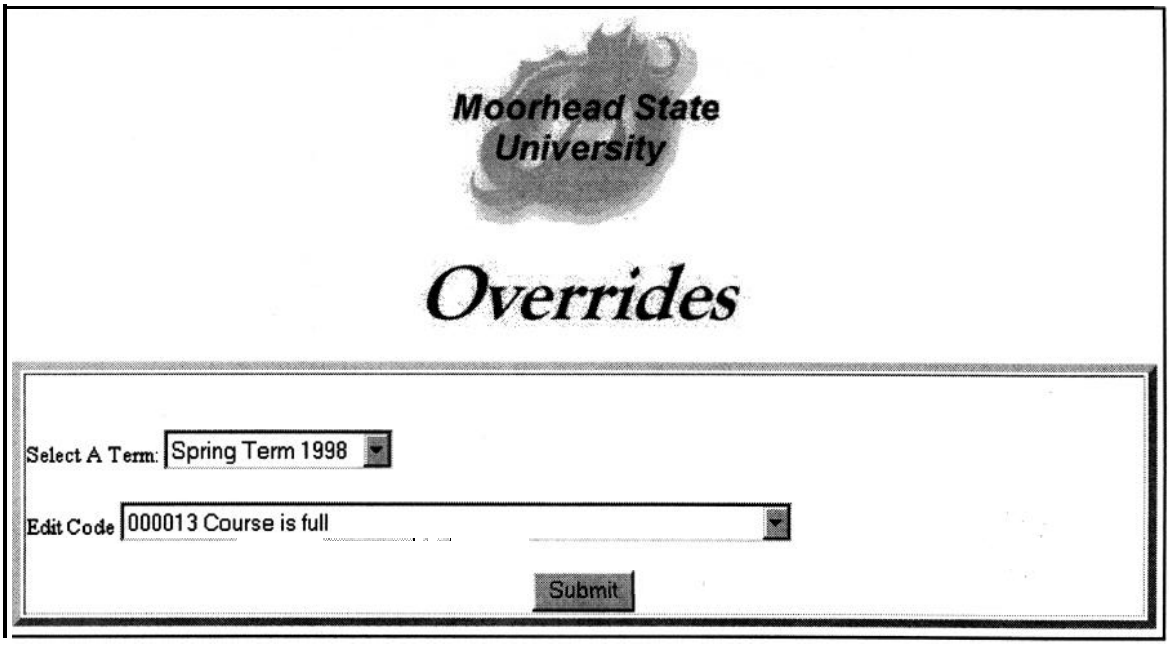
Number

General Ed Code

Registration Form

Add/Drop/Change	Course ID	Grading Method	Variable Credit
Add Course		Grade (01)	.
Add Course		Grade (01)	.
Add Course		Grade (01)	.
Add Course		Grade (01)	.
Add Course		Grade (01)	.
Add Course		Grade (01)	.
Add Course		Grade (01)	.

Figure 6: On-line Registration Form

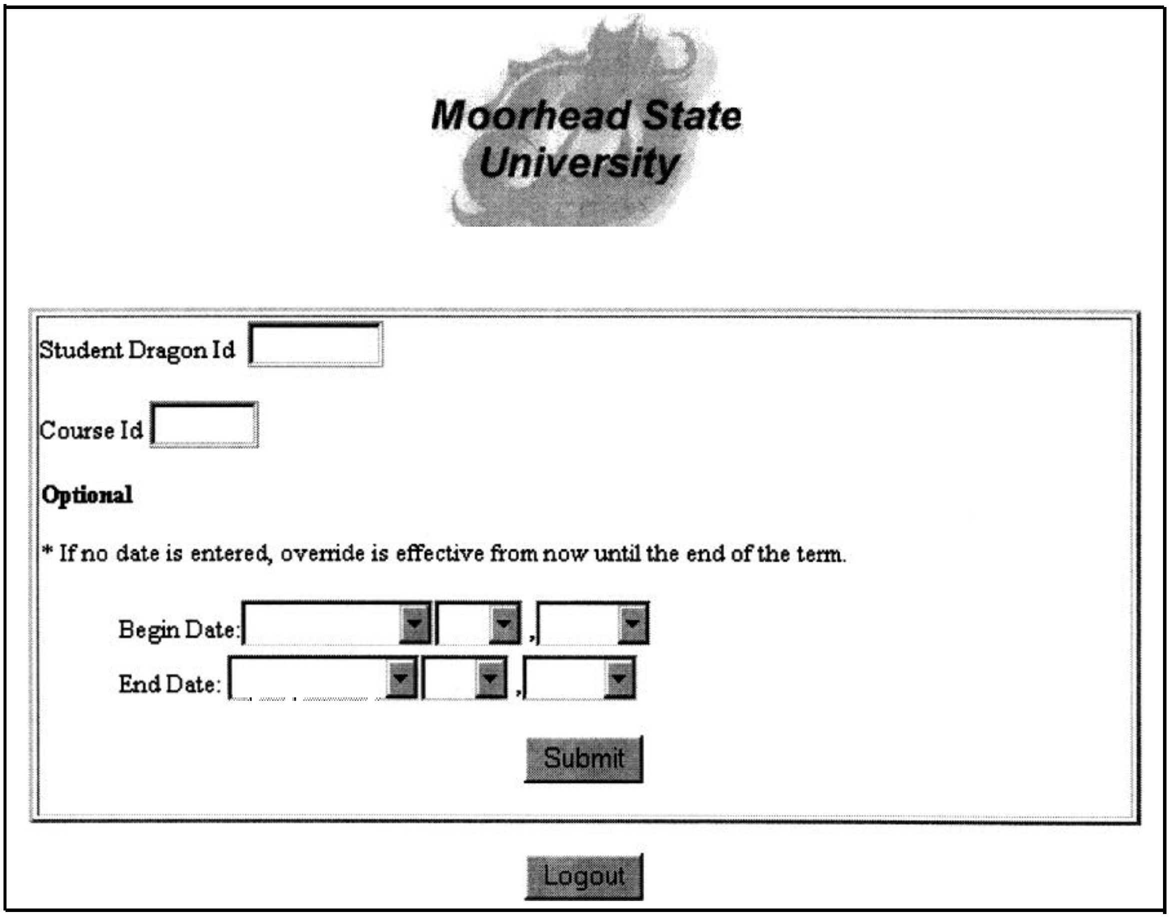


Overrides

Select A Term: Spring Term 1998

Edit Code 000013 Course is full

Figure 7: Faculty Override Selection Form



Student Dragon Id

Course Id

Optional

* If no date is entered, override is effective from now until the end of the term.

Begin Date: ,

End Date: ,

Figure 8: Faculty Override Information Form

Participation in NCSTRL: A Digital Library Experience

Dennis Amundson
Information Technology Services
North Dakota State University
Fargo, ND 58105-5164
amundson@badlands.nodak.edu

Abstract

The Computer Science Department at NDSU (NDSU-CS) has started to make its Technical Report (TR) series electronicly available through the WWW. Using a software package created by Cornell University (named Dienst), NDSU-CS is making its TRs available to a wide global audience through NCSTRL (pronounced “ancestral”), a world-wide distributed digital collection of CS technical reports. Users of NCSTRL can search through and view TRs originating from over 100 Ph.D.-granting Computer Science academic institutions through a seamless, hypertexted interface.

In this paper, an overview of NCSTRL will be given, as well as an overview of the problems that were encountered by the authors in preparing the TR database for use by NCSTRL. Usually, only hardcopies of the TRs existed, so a major portion of time has focused on digitizing the TRs into electronic formats that both Dienst and the Web population can handle. A quick discussion will point out current problems, as well as future prospects for electronic publishing at NDSU.

Flood! Recovering from a Natural Disaster of Epic Proportions!

Doris M. Bornhoeft
Computer Center
University of North Dakota
doris@sage.und.nodak.edu

Cathy Hilley
Computer Center
University of North Dakota

Abstract

The Spring of 1997 saw a major disaster in the form of a flood hit the Red River Valley area of North Dakota. The hardest hit area was Grand Forks, including the University of North Dakota (UND) campus which houses the administrative host for the North Dakota Higher Education Computing Network.

Volunteers were recruited to help move the host to NDSU Information Technology Services. Tapes were shipped off to bank vaults in small towns around the area. Staff were called in from numerous locations and worked in Fargo for weeks following the city-wide evacuation. This paper will describe the cooperative effort of UND, North Dakota State University (NDSU), and IBM to relocate the administrative host to the NDSU campus and restore computing in time to meet the North Dakota University System payroll on April 30th.

Flood! Recovering from a Natural Disaster of Epic Proportions!

Doris M. Bornhoeft
Computer Center
University of North Dakota
doris@sage.und.nodak.edu

Cathy Hilley
Computer Center
University of North Dakota

Introduction

Helicopters buzzed overhead as the agent entered the building. She headed straight into the machine room. "Eerie" she thought to herself - the room was absolutely silent - usually the noise hits you like a wall of sound as you walk in. Today -- no blast of cold air from the air conditioners, no sound of disk drives whirring in the distance -- just silence, ominous silence.

This may sound like the opening of a suspense novel or an episode of a popular television show but it's based on events that happened at the University of North Dakota. A year ago the "flood of the century" struck the Red River Valley and forced evacuation of a good portion of the cities of Grand Forks, ND and East Grand Forks, MN. This presentation will look at the relocation of the UND mainframe computing facilities the Spring of 1997.

First, a little background. The UND Computer Center houses the North host for the North Dakota Higher Education Computing Network (HECN). This IBM mainframe is responsible for administrative functions for the 11 universities and colleges that make up the HECN. These functions include student information, registration, personnel information, and payroll. The machine room is located in the basement of Upson II, a major concern when faced with a flood.

Evacuation

To aid the flood-fighting effort, UND had allowed release time the week of April 14, 1997 for sand-bagging and was closed on Friday, April 18th. A single operator remained on duty at the Computer Center to mount tapes, answer phones, and watch for flooding in Upson II. Cause for concern was the weakened storm sewer line that runs through the basement. Some Computer Center staff had already been evacuated from their homes.

The current UND Computer Center Disaster Recovery Plan calls for relocating the host to another location on the UND campus in event of a catastrophic disaster. The timeline presented in the plan has hardware installation within 7 days of the disaster, software and data restoration within 10 days, and processing to begin within 14 days. Needless to say, the other location on campus was not a viable relocation site. An additional site identified within Grand Forks was also flooded. The decision was then made to move the IBM host to NDSU in Fargo, 70 miles to the south.

Saturday, April 19th - The machine room was powered down, and the operator evacuated. A team of four IBM technicians arrived from Fargo and disconnected the most critical and/or expensive IBM hardware and moved it to the 3rd floor of Upson II. By this time, most staff (37 of 50 full-time employees) had scattered to homes of family or friends, to motels, or in some cases, headed out of the region. UND canceled the remainder of the Semester, 3 weeks early.

Sunday, April 20th - The remaining Computer Center staff and volunteers moved most of the remaining computer equipment from the machine room and student cluster in the basement to the third floor. Critical backup tapes were moved from the storage facility across the street from the Computer Center to the bank in Larimore, N.D. That evening the IBM and Unisys processors and peripherals were loaded into a volunteer's semi-trailer for relocation to North Dakota State University in Fargo the next day.

Monday, April 21st - The host was installed in a recently vacated computer room in the IACC (Industrial Agricultural Communications Center) building. In a conference room at the bank in Hatton, ND, recovery planning began: Primary goal - produce paychecks for the HECN's April 30th pay period; then, resume mainframe processing for the HECN. Even though UND's school year was cut short the other campuses were in the middle of Fall registration and final grades were fast approaching. Evacuated staff were located (many had reported in using the toll-free number UND had established for employees to call), and accommodations arranged for them to work out of NDSU. At UND, staff were cleared through security check points and escorted to Upson II where they disconnected more cables and sent them to NDSU. More than 8000 tapes were loaded into boxes and bags. Later that day and into the night staff and volunteers used space in the bank in Buxton to begin sorting the tapes back into some usable sequence.

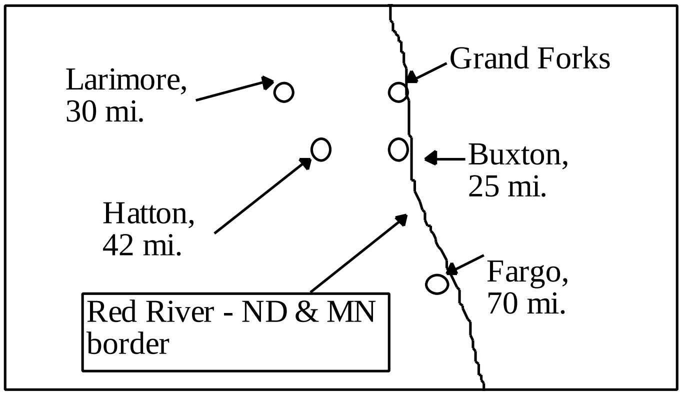


Figure 1: Locations used during the Evacuation

Tuesday, April 22nd - Re-installation of the hardware at NDSU began with Friday as the projected production day. NDSU welcomed UND staff with temporary office space, supplies, phones, workstations, and meal service. Alternative plans were developed for producing payroll at ISD (the State Information Systems Division) in Bismarck, if the UND system would not be available in time. Staff began rerouting communication lines to service the HECN through Fargo, a task made easier due to the statewide network (North Dakota Information Network - NDIN).

Wednesday, April 23rd - The IBM system was IPL'd by early afternoon. Testing of hardware and software ran through the night and into the next day.

Thursday, April 24th - Processing resumed for the HECN late Thursday afternoon, allowing normal payroll processing to proceed.

Fortunately, the basement of Upson II had remained dry, allowing staff to relocate the remaining UND Computer Center servers to the 3rd floor. These servers were brought back on-line Thursday. Included in these computers was the UNIX host running UNDInfo, the web-based campus information system. Staff arrived at NDSU and began updating the flood information web page using official UND press releases and updates sent via e-mail, FAX and telephone contacts with employees that remained at the UND campus.

Monday, April 28th - A help desk was set-up at NDSU and staffed by displaced UND employees. The last of the 11 campus' payrolls completed, continuing the tradition of not missing a payroll in 32 years. Planning for the return to Grand Forks began, the target move date, May 17th, to allow for completion of the May 15th payroll.

The Return Home

UND re-opened on May 8th. Staff in Grand Forks answered the phone and prepared for re-installing the equipment in the basement.

Saturday, May 17th the systems were powered down and readied for the move. Volunteers, assisted with moving tapes, forms, and miscellaneous equipment. By 1 p.m. the equipment arrived at Upson II. The IBM system was re-installed and IPL'd at about 7 p.m.; testing was conducted for the next four hours; and normal processing resumed by midnight.

Summary

The flood cost UND an estimated $43 million, 72 of 238 buildings were damaged, and 69 miles of underground steam tunnels and water, electrical, telecommunications and storm/sewer systems had extensive damage. The Computer Center lost approximately $50,000 worth of equipment, mostly communications equipment such as routers and converters that were located in flooded buildings. The IBM host did have damage due to the moving about but the replacement parts and repairs were covered by the maintenance agreement and did not result in any down time. Residence halls were re-opened as soon as basic services were restored to house a total of 1,960 members of the community (including faculty & staff), FEMA and SBA employees, and utility workers. The Hyslop Sports Center housed 4,955 volunteers representing approximately 400 different groups throughout the summer. Contrary to early predictions, UND's summer session started on time on May 12. This paper looks at relocation of the IBM mainframe system but many other computers were relocated during this time. For example, the Computer Center LAN team relocated their servers to the 3rd floor of Upson II to reinstate the Groupwise e-mail system and the School of Medicine relocated servers to Mayville State University . They all have a story to tell.

Despite our best laid plans, Nature has a way of throwing a monkey wrench in the works. The UND Computer Center Disaster Recovery Plan didn't allow for the possibility of the majority of the staff as well as the city to be evacuated. The stressful relocation came about so quickly and as smoothly as it did due to the major effort of the staff at UND and NDSU as well as a multitude of volunteers. A lot of progress has been made and the University and community continue in their recovery efforts and will do so for a long time.

Notebook Computers for Students:
Planning and Implementation on Two Campuses

Dr. Ray Brown
Vice President for Academic Affairs
Mayville State University and Valley City State University
Mayville, ND 58257
ray_brown@mail.masu.nodak.edu

Linda Brown
Information Technology Support Specialist
Mayville State University
Mayville, ND 58257
linda_brown@mail.masu.nodak.edu

Dr. Gary Hagen
Chief Information Officer
Mayville State University
Mayville, ND 58257
gary_hagen@mail.masu.nodak.edu

Joe Tykwinski
Chief Information Officer
Valley City State University
Valley City, ND 58072
joe_tykwinski@mail.vcsu.nodak.edu

ABSTRACT

INTRODUCTION

The paper offers a detailed picture of the experiences at Mayville State University and Valley City State University where notebook computers are provided to students and faculty. The rationale for the project and details of the planning process are included. A description of classroom renovations, networking, training, and assessment activities will be provided. Concluding remarks will summarize perceptions regarding opportunities and threats facing the campuses after several years of experience with required student use of notebook computers.

TECHNOLOGY PLANNING PROCESSES

The paper provides an overview of the detailed planning processes followed on both campuses. Faculty, staff and students successfully prepared the campuses for modification of classrooms, networking, professional development, and support of an expanded number of users.

CONCLUSION

Two university campuses decided to provide students and faculty with notebook computers. Planning and implementation processes were used to insure success. The lessons learned can be useful for other campuses.

Notebook Computers for Students:
Planning and Implementation on Two Campuses

Dr. Ray Brown
Vice President for Academic Affairs
Mayville State University and Valley City State University
Mayville, ND 58257
ray_brown@mail.masu.nodak.edu

Linda Brown
Information Technology Support Specialist
Mayville State University
Mayville, ND 58257
linda_brown@mail.masu.nodak.edu

Dr. Gary Hagen
Chief Information Officer
Mayville State University
Mayville, ND 58257
gary_hagen@mail.masu.nodak.edu

Joe Tykwinski
Chief Information Officer
Valley City State University
Valley City, ND 58072
joe_tykwinski@mail.vcsu.nodak.edu

Mayville State University (MaSU) and Valley City State University (VCSU) distributed notebook computers to faculty and students as part of a broader strategic commitment to create technology intensive campuses. This paper provides background information, implementation details, and lessons that the campuses learned.

RATIONALE

The needs of learners were at the center of conversations throughout the entire process of determining strategies for enhancing use of technology. There was a firm belief in the critical importance of the notebook computer initiative. With messages from employers and alumni indicating that it was imperative to significantly increase student experiences with technology, the campuses felt that significant changes were needed.

If a university believes that enhanced technology literacy must be provided for all students, then some sort of universal computer model emerges as a clear goal. A key component of campus deliberations centered on the benefits which students and faculty gain from enhanced communication that is not constrained by time and space.

A number of specific benefits for campuses with universal notebook computer access can be identified:

- Students have 24-hour access to networked institutional resources and the Internet from their dorms or apartments.
- Students possess 24-hour access to powerful software applications. These software packages are standardized across the campuses altering the model for computing support. Everyone on campus serves as potential resources for answering questions.
- Students, faculty, and staff are able to experience benefits from communication and efficiencies from relationships that are not dependent on time and location.
- Students are assured access to professional positions with relatively high salaries/benefits.
- An intensive computing environment is the only way institutions are able to assure that all students are graduating with appropriate technology skills. Students completing only one general education computer literacy course will graduate four years later with skills that are severely limited.
- Portfolios and a number of other learning innovations are essentially impractical across an entire institution without a common technology base.
- Universal access to laptops permits development of courses for delivery over networks and positions the institutions for delivery of instruction to dispersed audiences.
- E-mail permits enhanced communication between faculty and students.
- The reputations of the institutions and graduates are enhanced.
- The traditional campus computing model is frustrating for faculty and students who desire greater use of technology. There is uneven use of technology at many institutions based on decisions made in the past and the aggressiveness of specific faculty members who were able to secure access to labs. Computer labs on most campuses are essentially at maximum capacity use throughout the day for classes and students have difficulty finding time for use of labs to complete assignments. This situation is frustrating for staff who find it difficult to maintain labs and computers in offices. There are too many different hardware and software combinations on most campuses.

- Most institutions are finding it very difficult to acquire sufficient dollars for replacements/upgrades of existing computer facilities.
- Reallocation of resource expenditures in recent years to purchase computers and networking is not easy to sustain for much long periods of time. Needs for use of equipment dollars in areas beyond computers remain.
- The campuses have been aggressive and successful in recent years using grant writing as a strategy for securing funds to purchase technology and support faculty/staff development. These efforts will continue, but it is likely that campuses will face more competition in the future from other institutions and the odds of maintaining a record of success are uncertain.

CONTEXT FOR THE INITIATIVE

MaSU and VCSU are separated by 75 miles and share significant similarities in size, history, and academic programs. After a period of study and public input, the State Board of Higher Education decided to join the two campuses under a single administrative team.

The SBHE released a strategic academic plan in April 1995 that provided mission differentiation for MaSU/VCSU with the statement, "Through institutional partnership, serve as a model of multi-campus administration by a common administrative team and provide learner-centered instructional innovation with an emphasis on collaboration…"

Another document issued the same month by Chancellor LarryIsaak, provided further clarification of the partnership within the North Dakota University System, "Consistent with past mandates from the State Board of Higher Education and with the University System's academic strategic plan, Valley City State University and Mayville State University are directed to collaborate, including other interested campuses and high schools, to improve teaching with technology and share the results with other campuses." The notebook initiative and related innovations are consistent with this vision.

EARLY PLANNING

A group of faculty and staff from both MaSU and VCSU traveled to the University of Minnesota-Crookston (UMC) during the 1993-94 academic year to observe their notebook initiative. UMC was the first university in the nation to distribute notebook computers to all full-time students and faculty. The initiative soon became known as ThinkPad U. The group on the field trip returned to the respective campuses and began conversations with others. During the 1994-95 academic year, a group of faculty, staff and students at VCSU formed a Technology Advisory Committee (TAC). A similar Technology Planning Committee (TPC) at MaSU was formed. Both campuses used a planning model developed by the Center for Innovation in Instruction (CII). The CII provides technology training and assistance in North Dakota. The VCSU TAC developed a technology plan which included a recommendation to the administration that serious thought be give to the notebook concept.

The administration discussed the options and entered the 1995-96 academic year with a charge to various campus constituencies for further study and provided a deadline of September 15, 1995. By this date, faculty, students, and staff would provide a recommendation for the two campus Executive Team to proceed with the notebook initiative in Fall 1996, drop the idea, or postpone the implementation until Fall 1997.

Very few individuals expressed complete opposition to the notebook concept. Students at both MaSU and VCSU were evenly divided between the two dates. Roughly two-thirds of VCSU faculty supported the earlier target date. MaSU faculty preferred at Fall 1997 starting date. The Executive Team met and decided to launch a two campus effort collectively known by the initials ILT for Improving Learning with Technology.

IMPLEMENTATION

The North Dakota State Board of Higher Education approved a request from MaSU and VCSU at the November 1995 meeting to charge a $475 technology fee each semester. The full fee would be paid by full time students with a prorated per credit hour fee for part time students. An approximate breakdown of the technology fee follows:

Fee	Use	Comments
$378	Computer Lease	Includes funding for extra computers for replacement of computers returned for repair and an Ethernet card
$5	Case	paid for a padded carrying case at VCSU; case provided by the vendor at MaSU for no additional charge
$9	Insurance	Students pay a $750 deductible for repairs or losses not covered by the warranty. These funds are used to compensate the institution for additional losses.
$35	Software	Purchase, maintenance and upgrades
$30	Network	Repay a 10 year bond issued with proceeds for networking.
$18	HelpDesk	Hire staff and equip the Help Desks
$5	Printing	Paper and printing supplies

With SBHE approval, the level of planning activity increased. A joint MaSU/VCSU committee researched hardware and software options. A new Chief Information Officer (CIO) position was created at VCSU and computer staff were reorganized with a reporting relationship to the VPAA. At MaSU the reorganization and change in reporting relationship took place at the same time, although identification of a MaSU CIO was delayed until the Spring semester 1997.

Help Desk Directors were hired on each campus in the spring of 1996. Detailed planning, including preparation of a RFP for vendors was organized. Notebook computers were distributed to all faculty at VCSU in January 1996. Ten notebooks were acquired at MaSU to support early training.

In August 1996 notebooks were provided to all full time VCSU students and to all full time MaSU faculty members. An extensive array of training opportunities were organized for the specific software suite, e-mail and multimedia software. Classroom remodeling and networking of the campuses was completed at VCSU with preparation for remodeling at MaSU. MaSU students received their notebook computers during the Summer and Fall semesters of 1997.

IMPLEMENTATION TIMELINES

Each campus followed a somewhat similar timeline prior to the start of the notebook computer initiative.

Mayville State University

1990	MaSU adopts Reflective Experiential Knowledge Base
1992	Bush Foundation Cooperative Learning grant received SBHE authorizes Partnership with VCSU

1994 Field trips to Alverno College and UM-Crookston

1995 SBHE approves Technology Intensive Campus Initiative with notebook computers

Title III grant received; coordinator hired; equipment purchased for multimedia lab

1996 Initial notebooks arrive on campus; training is started for faculty and staff

John L. Blackburn Award received from American Association of University Administrators for Exemplary Model of Administrative Leadership

Notebooks provided to all faculty at start of fall semester

1997 Campus wiring initiated

RFP issued for notebook initiative

IBM ThinkPad 365's distributed to all summer school students

Red River Learning Corridor is created

1998 Cross Training begins with VCSU

Valley City State University

1989 Ad hoc Center for Innovation Committee begins meeting

1990 VCSU receives mission designation from SBHE for statewide leadership in effective use of instructional technology

1991 Bush Foundation planning grant received to fund travel to exemplary sites (Dakota State University; Northwest Missouri State University, University of Nebraska, Chadron State University) and consultant visits

1992 Trip to TIE Conference in South Dakota

SBHE authorizes Partnership with MaSU

1993 Initial funding from ND ETC for Center for Innovation in Instruction

Approval of SBHE for Instructional Technology major

FIPSE grant received for development of innovative instructional strategies

1994 Field trips to Alverno College and UM-Crookston

Kathryn Center Adventure Learning Course used with entering students

1995 SBHE approves Technology Intensive Campus Initiative with notebook computers

Title III grant received; coordinator hired; equipment purchased for technology lab

RFP issued for notebook initiative

1996 Campus wiring initiated

IBM Thinkpad 365's provided to faculty; training is started for faculty and staff

Classroom renovation begins

John L. Blackburn Award received from American Association of University Administrators for Exemplary Model of Administrative Leadership

IBM Thinkpad 701CS's provided to all students at start of fall semester

1997 Students return to campus several day early for intensive orientation

Red River Learning Corridor is created

1998 IBM Thinkpad 380's distributed to all students

Cross Training begins with MaSU

THINGS TO CONSIDER

There are some specific items that should be considered by campuses when planning the notebook computer model.

- Administrators should be visible and enthusiastic supporters. Plan for use of the technology as your primary communication channel, the server as a repository for documents, and your web site as the place to find catalogs, schedules and other important information.
- Make an institutional commitment. Standardize software and hardware.
- Provide an extended period for campus discussion and planning.
- Repeated trips to campuses with notebook initiatives is very helpful.
- Provide meaningful roles for students, staff and faculty as full partners in decision making.
- Identify someone as CIO and centralize responsibility for campus technology.
- It is helpful if a faculty technology leader is willing to serve as CIO.
- Create a Help Desk to serve as a one-stop technology support center.
- Upgrade software on faculty and staff desktops as early as possible to match what you expect to load on the notebooks.
- Begin the distribution as a trial run for students during the summer session prior to the academic year in which you plan to start the initiative.
- Close down traditional computer labs at the same time notebook computers are distributed to students.
- Provide extensive opportunities for training. Include both faculty and staff in training to gain staff support and understanding for the initiative.
- Use every opportunity to communicate with prospective students and their parents.
- Maintain good communication with people in your broader community.
- Develop a dialogue with students and faculty about the implications of the technology.

FACULTY AND STAFF DEVELOPMENT

Faculty and staff, working to focus attention on learner-centered education, gained over $2 million in competitive grants for professional development. This effort included a major federal Title III (Strengthening Developing Institutions) collaborative grant for $1.7 million, one of only 7 percent of proposals to receive funding in 1995. The five year project establishes parallel, but intersecting and mutually reinforcing trajectories to further enhance the transformation of the MaSU and VCSU into learning-centered institutions.

The Title III project builds on the strengths of each university. Faculty at VCSU are working initially to define learning outcomes and helping students build CD-ROM based portfolios documenting their learning progress. MaSU faculty are receiving intensive training in multimedia technologies and are building on their knowledge in effective teaching and learning strategies. Representative faculty teams on each campus will assist with dissemination of new capabilities at their home institution and with faculty from the partner campus. Since the start of the project, almost 90% of full-time and 60% of part-time faculty participated in one or more of a series of workshops on specific software packages and use of the notebook computers.

Significant thought was devoted to delivery of training for faculty, staff and students. The 3 credit General Education computer literacy course was redesigned as the primary training vehicle for all new students. A new course, Microcomputer Software and Network Resources (1 Credit), was created at VCSU for upper class students who had completed the Introduction to CIS course in traditional lab setting using Windows 3.1. At MaSU, Windows95 and the Office software suite were installed in computer labs over one year ahead of the notebook computer distribution. This action significantly minimized the problem of providing training for upper class students. A large number of training sessions on various topics were provided over the summer and an intense training period was developed just before the start of the first semester during the faculty pre-service.

RFP PROCESS

The campuses also paid considerable attention to the process employed for selecting a hardware vendor. A Request for Proposals (RFP's) was drawn up and distributed to a large number of notebook manufacturing firms and other vendors. The RFP laid out specific details regarding the anticipated timeline for implementation. In addition, detailed information about expected hardware and software capabilities was included. It was assumed on the campuses that the vendor selected would be expected to provide attention to the notebook computer initiative. Other parts of the RFP requested pricing and leasing information for hardware and software. Vendors were also asked to provide assistance with service and marketing of the notebook efforts.

Following the mailing of theRFPs, committees on each campus considered vendor responses. Selected finalists provided sample computers matching the specifications offered in the respective vendor response. These sample notebook computers were then displayed on campus permitting testing by students and faculty. Feedback was collected by the selection committees as additional input for decision making.

The RFPs requested and the various vendors offered alternative packages. Through the separate campus processes, IBM surfaced as the selected vendor. The campuses then began a process for establishing a more formal partnership.

The following models of notebook computers are in use: IBM Thinkpad 365CSD's (distributed to faculty at VCSU January 1996 with 12-24MB RAM, 540MB HD, CD-ROM, DX4/75 processor); IBM Thinkpad 365ED's (distributed to faculty at MaSU August 1996 with 24 MB RAM, 540 MB HD, CD-ROM and distributed to MaSU students Summer and Fall Semesters 1997 with Pentium processors, CD-ROM drives, 24 MB RAM, 1.08 GB HD); and, IBM Thinkpad 380 notebooks (with Pentium processors and built-in CD-ROM and floppy drives were provided to VCSU students during January 1998 following a year and one-half with the original IBMThinkpad 701CS's distributed 1996 with 12MB RAM, 720MB HD, 486 DX4/75 processors).

The standard software selected includes: Windows 95, Microsoft Office Professional, Novell GroupWise, anti-virus software, and an Internet browser.

Costs associated with acquisition of the notebooks, cases, and modems were split between faculty computers and student computers. Reallocation of state funds was used to finance a 30 month hardware lease to acquire the faculty computers. Student fees provide the funding for a 36 month hardware lease and a 24 month software/case/card lease.

Extensive classroom renovations were completed at VCSU during the 1996 summer period and at MaSU during the 1997 summer period. Classroom equipment typically includes a projection device or television/monitor, scan converter, VCR, overhead camera, videodisk player, and printer. Furnishings include tables with electrical outlets and Ethernet connections, and chairs.

Eighteen MaSU classrooms and twenty-five VCSU classrooms were equipped for video projection at a per room cost ranging from $1,000 - $4,000. The campus networks include an Ethernet/NovellNetware environment. Fiber optic backbones were installed with a T1 line to modem pools that provide dial-in access for faculty, staff and students. Networking was completed to student dorm rooms at MaSU, while clusters of network jacks provide access to VCSU students on each dorm floor. Resident hall wiring was paid for by auxiliary services. Campus and classroom wiring was covered by proceeds from a state bond and with appropriated state funds. Local funds were important for key activities which were not covered by other sources.

Following wiring of the multimedia classrooms, the traditional computer labs were dismantled. This was timed to coincide with the distribution of notebook computers. Once students had access to the campus network, there was no longer a need to maintain labs.

There was some argument on each campus to retain labs for a transition period. In retrospect, the decision to close the labs was a significant step and a strong signal that our campuses were, indeed, notebook campuses. Further, without labs, everyone on campus was pushed to use the new technology. Closing the labs also significantly reduced a major demand on computer staff time and permitted redistribution of desktops from the labs to staff offices. All of the desktops that were capable of running the campus standard software were used. Many less powerful machines are in use as file or print servers.

All staff members, including physical plant employees and others who might not normally expect to access a computer at work, can now use the campus network. This proved to be a very popular and empowering move for people who might not otherwise have felt that they were a part of the technology initiative. It was also a necessary step, since e-mail rapidly emerged as the main vehicle for campus communication. It also provided every employee with convenient access to the WWW where they could access both campus and NDUS employee manuals and other job related documents.

Arrival of notebooks and planning for preparation of software images for downloading was the next major consideration. A great deal of time was spent on each campus determining the ideal configuration of the campus standard software. This was done with the hope that the final image would both ease student use of the software, but also minimize demands for support. The process of downloading the image to hundreds of hard drives improved significantly over time. The initial process depended on the DOS backup/restore command. Everyone recognized that this was ineffective and time consuming.   PK-Zip was used for a time to compress the images, shortening time for downloads. Finally, licenses for Ghost software were acquired to permit even faster downloads to multiple computers at the same time. Creation and downloading of the notebook image is a responsibility of the respectiveHelpDesks and the task can now be completed quite efficiently.

The HelpDesk was also assigned the task of maintaining an inventory of the notebook computers. Each student is provided with a specific computer and is responsible for that computer until it is returned to the HelpDesk. Students attend a short “care and feeding of the notebook” class and sign a detailed contract agreement before receiving their computer. This has proven to be a relatively effective process for passing along a great deal of information to students. At the same time, emphasis is placed on student responsibility for protecting the computers. To date, there have been almost no problems arising from uncaring students.

What’s it like to be a Notebook campus?

The notebook computers have definitely changed the teaching and learning process. Additionally, the project has pushed faculty and staff to reconsider traditional routines and communication patterns. Students, both in the classroom and as student campus employees, are making significant contributions to greater workplace efficiency with better understanding and use of the productivity software that is a standard across the campus.

There are some on-going support issues which are altered by the shift to notebook computers. Some of these issues are common across all higher education institutions. For example, use of the WWW and the campus network are growing rapidly. Dial-in access for off-campus students was a problem early in the project. Completion of dorm wiring, emergence of a local ISP, and installation of additional modems in the computer centers has made dial-in access much easier.

Ease of network printing was a major goal on each campus. It took a while for students and staff to realize that they did not need their own individual printer. A parallel effort to encourage people to share and store documents electronically has kept the cost of stocking toner and paper to manageable levels. Everyone is becoming much more adept at placing files on the file servers for sharing or later retrieval. A very powerful use of the network is also emerging since HelpDesk and computer center staff members can now perform “unattended installs” as a way to upgrade software easily.

From time to time, viruses present problems since they can be spread quite rapidly through one infected attachment that is addressed to all students, faculty and staff. In general though, viruses do not require the constant attention of staff which was true in the old lab setting with students carrying infected floppy disks around from one desktop to another. Virus detection software has also become much more useful and user friendly in recent years.

SWOT ANALYSIS

A SWOT analysis was performed as part of the technology planning process by personnel on the campuses during the 1996-97 academic year. A summary follows:

Strengths

- The universal notebook computer model ensures access to modern technology with constant and scheduled updating. It is easier to conduct long range technology planning.
- Universal access to a standard platform creates efficiencies and synergy in distributing information, communicating, training and providing technical support.
- Widespread use of notebook computing meets the needs of learners with any time, any place access to technology and information.
- Universal access to notebook computers promotes active learning and active learning is linked by researchers to increases in knowledge gains, retention, and synthesis.
- All disciplines have access to computers and networked information.

Weaknesses

- The model requires all students to pay higher fees.
- Part-time students check out a computer from the Help Desk before each class or they can choose to pay the full technology fee and have a computer issued to them on a full-time basis.
- These payment options are not always practical for students taking courses off campus. The fee can be waived, however, this does not solve the access issue if students do not have their own modern personal computer and software.
- The cost of technology is listed as a separate fee on student fee statements. Unlike the cost of building maintenance, faculty and staff salaries and other items associated with instruction, the cost of technology is set up for individual scrutiny.
- Limited resources and a desire to capitalize on the notebook computers results in most of the information technology resources being allocated to needs related to the model, leaving reduced resources available for individual requests.

Opportunities

- Assessment models based on the demonstrated skill development of each learner are feasible when students use their computers to record and store documentation and faculty use network connections to retrieve these documents for evaluation.
- Learning modules accessible via the Internet make customized learning feasible. Customization can include content to be learned, time and place of learning, and the pace of learning.
- The ease of communicating with anyone, anywhere, creates new opportunities for partnerships and information sharing.

Threats

- New legislative mandates require long range technology planning and tie institutions to a long range plan may become rigid and unresponsive to customer needs. The institution may not be able to take advantage of emerging technologies and time-sensitive purchasing opportunities that proliferate in the technology market.
- The institutional lease of student notebook computers is structured for a fixed number of computers for a fixed time period of 3 years. If enrollment drops, the institution must find alternative funding mechanisms to cover the lease.
- Higher costs to students may reduce enrollments if students and/or parents do not perceive or understand the added value.
- The model is based on using student fees collected each semester to pay for the institutional lease of notebook computers. Legislative restrictions on lease agreement terms, as proposed during the 1997 legislative session, could kill the model entirely.

LESSONS LEARNED AS NOTEBOOK CAMPUSES

There are a number of lessons gained through our early experiences:

- Take time to recognize and celebrate use of the technology.
- A commitment to notebook computers will change teaching and learning. You will not be able reorganize fast enough.
- Do it all at once.
- Standards, standards, standards - it's the only way to survive.
- It's the network…stupid.
- There is not an ideal model for use of notebook computers by part time students and part time faculty.
- Administrators need to explicitly think about the appropriate level of encouragement/force. Hold policy exceptions to an absolute minimum.
- You will find that both faculty and staff will be reflecting on and changing their concept of "best practice"
- Do not set your notebook and technology fees too low. If possible, build the price of the notebooks into regular tuition.

FINAL COMMENTS

After half a decade of conversation about computers at MaSU and VCSU strong support for the notebook computer model remains. Students and faculty possess powerful tools for learning. Notebook computer initiatives do not solve all campus technology problems. Old issues are replaced with new and different issues. The model does, however, provide a path for significant increases in campus technology use that is manageable on small campuses with limited resources.

Providing Universal Student Access to Technology:
A Summary of Alternative Models

Dr. Ray Brown
Vice President for Academic Affairs
Mayville State University and Valley City State University
Mayville, ND 58257
ray_brown@mail.masu.nodak.edu

Dr. Phillip Heeler
Professor and Chair
Department of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
pheeler@acad.nwmissouri.edu

Dr. Roger Von Holzen
Assistant Professor
Department of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
0100010@acad.nwmissouri.edu

ABSTRACT

INTRODUCTION

A growing number of universities are creating technology intensive campuses with universal requirements for computer ownership. The paper provides an overview of implementation strategies on selected campuses. A matrix is included with descriptive details for universities with universal student access strategies. The universities express a commitment for minimum hardware and software standards. A number of distinct implementation models are evident. An analysis of alternative paths pursued by the respective campuses helps clarify common themes and distinctive features.

CONCLUSION

Faculty and staff on many campuses are working to improve student access to technology. Most of these initiatives rest on a belief that students will need better skills in use of computers. While each campus community necessarily pursues goals in ways which reflect unique cultures and missions, there are common models for creating technology intensive institutions which can be identified and compared.

Providing Universal Student Access to Technology:
A Summary of Alternative Models

Dr. Ray Brown
Vice President for Academic Affairs
Mayville State University and Valley City State University
Mayville, ND 58257
ray_brown@mail.masu.nodak.edu

Dr. Phillip Heeler
Professor and Chair
Department of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
pheeler@acad.nwmissouri.edu

Dr. Roger Von Holzen
Assistant Professor
Department of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
0100010@acad.nwmissouri.edu

INTRODUCTION

A growing number of universities are creating technology intensive campuses with universal requirements for computer ownership. The paper provides an overview of implementation strategies on selected campuses. A number of distinct implementation models are evident and an analysis of alternative paths pursued by the respective campuses clarifies common themes and distinctive features.

In the 1980's, Clarkson University (Potsdam, NY), Bentley College (Waltham, MA), Drew University (Madison, NJ), and the New Jersey Institute of Technology (Newark, NJ) were among the first institutions to either require purchase of computers or provided desktop computer to entering students. Northwest Missouri State University (Maryville, MO) stands out among early leaders with full implementation of a terminal-based Electronic Campus in the fall 1987.

By the 1990's a number of these campuses made a transition from desktop to notebook computers. In 1993, the University of Minnesota-Crookston (UM-C) was the first institution to distribute the same model of notebook computer to all students. Valley City State University, Waldorf College, and Mayville State University followed in 1996 and 1997.

UNIVERSAL ACCESS TO TECHNOLOGY

RATIONALE

An increasing number of higher education institutions are implementing requirements for student access to computers. A number of considerations are driving campuses to consider the possibility of universal access:

- Rapid changes in hardware and software require significant reallocations of resources or new sources of revenues if a campus hopes to expand the use of technology. A universal technology requirement allows students to consider payment of increased fees through the financial aid process.

- Changes in the workplace are raising the minimum entry-level standard for technology skills and premiums exist for graduates with networking and programming backgrounds. It is no longer considered acceptable to restrict access to certain majors and their faculty. If it's important, then all students and faculty expect to have access to needed learning resources and communication channels.

- The current model of labs is increasingly difficulty to sustain. Institutions are finding maintenance of labs and upgrades for software to be a major task. More faculty are trying to schedule class time in computer labs and students are finding it more difficult to use campus labs for school related work.

- Service and support is becoming a nightmare on many campuses. Though staffing levels are increasing, it is not uncommon to find campus expectations continuing to rise more rapidly. In addition to the traditional support for labs and offices, computer centers now must deal with campus networks and modem pools. Universal standards significantly increase efficiency of service, support and training.

- Access to computer networks and the related enhancement of communication are major factors pushing institutions to consider some sort of universal access. Faculty require easy access in order to maintain professional contacts. Students expect to have e-mail communication with their families and friends. Use of world wide web resources is rising dramatically and 24-hour access is expected.

- Mobility of computing resources is an important consideration for institutions with notebook computers as the campus standard. This benefits students both on-campus and off-campus. Students carry their computer with them from one classroom to another. The institution does not need to provide multiple dispersed labs for a given group of students to use in various classes. Notebook computers also fit nicely with educational philosophies that emphasize more active and community-based experiential learning opportunities. From student teachers to business interns to students involved in service learning and field-based research, notebook computers can follow along with students.

- Institutions can alter funding strategies. Rather than depend on irregular one time capital expenditures to finance technology purchases, there is an opportunity to build technology expenditures into the budget as an operating expense.

- Technology use is integrated throughout the entire campus and curriculum. Students can graduate with impressive levels of skills and sophisticated appreciation for the power of technology. In an already crowded curriculum, there is no way to add enough courses to achieve similar levels of proficiency.

- Universal software and hardware standards permit campuses to consider assessment strategies which would otherwise be difficult or impractical to implement.

ALTERNATIVE MODELS

In Resmer [Resmer et al. 1995] three models for universal access programs are identified: the Textbook model, the Department-oriented, multi-level model, and the Single vendor/machine model. The Textbook model refers to situations where the student makes the purchasing decisions. Students have flexibility in terms of purchase and may decide to share computing resources with other students. In the Departmental model, decisions are based on departmental needs rather than institutional needs. With the Single vendor/machine model, an institution works with one firm and requires all students to use the same computer.

It is important to note that institutions may alter their programs for universal access as time passes. Drew University began a Computer Initiative in 1984 with a requirement for desktop computers. Notebook computers were specified as the campus standard at Drew in 1988.

Bentley College began issuing desktops to all freshmen in 1985. Four years later, students were given an option to either rent or buy their computers. In 1995, students were required to purchase their computers.

Northwest Missouri State University became the first all Electronic Campus in 1987 with Digital Equipment text-based terminals in every dorm room and office. The terminals were connected by a campus network to VAX mainframe computers. After a pilot notebook computer initiative known as EC+, a decision was made to replace the terminals with desktop computers for the 1997-98 academic year.

Resmer, et al.	**Brown, Heeler, Von Holzen**
1. Textbook model	1. Required Minimum Standard
2. Department-oriented, multi-level model	2. Computers Provided by Program
3. Single vendor/machine model	3. Desktops Provided to All Students 4. Notebooks Provided to Entering Freshmen 5. Notebooks for all Students

This paper offers five models. The first two, Required Minimum Standard and Computers Provided by Program, are equivalent to theResmer Textbook and Department-oriented models. The last three (Desktops Provided to all Students, Notebook Provided to Entering Students, and, Notebooks Provided to all Students) are variations of the finalResmer model (Single vendor/machine model).

The distinctions proposed in this paper reflect different philosophies of the institutions. There are interesting implications for planning, learning, and support that follow a particular decision.

Required Minimum Standard

Some institutions set minimum levels of computing capability through agreed upon hardware and software standards. In some cases, the hardware standard is a specified model of notebook computer. More often the standard is defined as a minimum level of microprocessor, RAM, and hard drive space. The institution may provide students with the option to purchase computers through an on-campus source like the bookstore. The institution may encourage students to purchase their computers from vendors in the community or through mail-order firms. Six of the eight campuses following this model define desktop computer standards. The standard can be a specific brand as in the case of Dartmouth College with a recommendation that students in the 1997-98 academic year purchase an Apple Performa 6400/180 computer with 32 MB RAM and CD-ROM. Drexel University requires students to have “personal access to a computer” and recommends either a Windows machine with 32 MB RAM, 166 MHz processor and 8X CD-ROM or the Mac equivalent.

Desktops Provided to all Students

A number of institutions provide students with specific models of desktop computers. The key distinction offered between this model and a similar notebook computer requirement rests on price and performance issues. At this point in time, desktop computers typically offer faster microprocessors and more memory than equivalently priced notebook computers. There is also an issue of peripherals. It is usually easier and cheaper to install extra memory or special function cards in desktop machines.

The five U.S. service academies all follow this model. In addition, it is more often followed by more specialized institutions like New Jersey Institute of Technology and Stevens Institute of Technology with students majoring in engineering or technical fields. Another institution following this model is Wesleyan College in Macon, Georgia. Wesleyan started a Computer Focus Program in 1989 with distribution of Mac desktops to new entering students.

Notebooks Provided by Program

Faculty in a specific program like business administration might decide that students need universal access to computers. For political reasons the faculty in a particular school or college may decide to proceed rather than wait to convince their peers or they may recognize that programs differ in their computer needs. This model is common among two year institutions like Northwest Technical Colleges (MN) where computer use may be more logical or critical for some programs, but not for others. The model also is used by large institutions where a campus wide implementation would be difficult. Columbia University Business School began requiring M.B.A. students to purchase notebook computers in Fall 1992. The University of Oklahoma College of Engineering specifies that students select from one of two Compaq portable lines or IBM's model ThinkPad 380. The University of Missouri College of Education currently requires majors to use MacPowerbooks.

Notebooks Provided to Entering Freshmen

In this model, institutions often identify a single notebook computer for each year. With each new entering class of freshmen, a new model is selected. After a four year period, all students would possess notebook computers. Typically, the students retain use of the computer for their entire four year academic career. They are often also permitted to take the computer following graduation. In some cases, faculty are required to also use notebooks. More frequently however, faculty are given an option to use either desktops or notebooks.

This model appears to be the most common with twelve institutions making commitments to become "notebook universities". Wake Forest University and Seton Hall may be the best know institutions following this path.

Notebooks for all Students

Some institutions decide to select one notebook computer and distribute the specific computer to all students. These institutions intend to use the computers for two or three years before replacing all student computers at the same time. While there may be options for students to purchase computers at graduation, this is typically promised at "fair market value" and students receive no benefit from accumulated payments over time. UM-C was the first campus to follow this model. Three or four years passed before Valley City State University and Mayville State University in North Dakota, Waldorf College in Iowa, and Clayton College and Floyd College in Georgia joined the ranks of notebook campuses during the 1997-98 academic year.

FIVE ALTERNATIVE MODELS FOR PROVIDING UNIVERSAL ACCESS

1. Required Minimum Standard

- Bentley College (n)
- Clarkson University (d)
- Dartmouth College (d)
- Drexel University (d)
- Georgia Tech University (d)
- Sonoma State University (d)
- VPI Colleges of Business & Engineering (d)
- Villanova University College of Commerce & Finance (n)

n=notebook standard
d=desktop standard

2. Desktops Provided to all Students

- Northwest Missouri State University
- New Jersey Institute of Technology
- U.S. Air Force Academy
- U.S. Coast Guard Academy
- U.S. Merchant Marine Academy
- U.S. Military Academy
- U.S. Naval Academy
- Wesleyan College

3. Notebooks Provided by Program

- Columbia University Business School
- Columbus State Community College
- North Dakota State University Department of Architecture & Landscape Architecture
- Northwest Technical Colleges
- University of Oklahoma College of Engineering
- Sheridan College
- University of Missouri College of Education
- Vermont College of Norwich University

4. Notebooks Provided to Entering Freshmen

- Acadia College (1996)
- Drew University (1988)
- George Fox University (1991)
- Grove City College (1994)
- Hartwick College (1993)
- Houghton College (1997)
- Nicholas College
- Rose-Hulman Institute of Technology (1995)
- Sacred Heart University (1995)
- Saint Gregory's College (1997)
- Seton Hall University (1998)
- Wake Forest University (1996)
- West Virginia Wesleyan (1997)

5. Notebooks for all Students

- Clayton College (1997)
- Concordia College-St. Paul (1998)
- Floyd College (1997)
- Mayville State University (1997)
- University of Minnesota-Crookston (1993)
- Valley City State University (1996)
- Waldorf College (1996)

# represents the first
year of notebook distribution

IMPLICATIONS FOR INSTITUTIONS

In spite of many benefits, universal technology requirements can involve some costs or risks for both students and institutions. Costs and benefits vary to some extent across the implementation models outlined in this paper.

In general, there is a direct increased cost to students and parents. Either the student needs to purchase a computer or there is a higher fee paid to the institution. In a traditional setting, institutions struggle to acquire resources through higher tuition and/or increased state appropriations to fund technology purchases for labs and employee use. As described earlier, inability to provide sufficient access has led many students to also purchase computers for their private use.

While a case can be made that students benefit from increased technology access, irrespective of the nature or extent of faculty use, there are certainly higher anticipated benefits as the teaching/learning process is reevaluated to take advantage of technological capabilities.

Required Minimum Standard

In some sense, setting a required minimum standard probably carries the lowest risk to an institution in the short term. Faculty are able to design courses based on assumptions about student technology capability. It is the student's responsibility to find technology resources.

There are however potential pitfalls. There is the potential for frustration among both faculty and students, depending on how aggressive the institution monitors student access. Even with specified standards for processors and memory, there can be significant differences both within and among brands of computers. These differences can result in significant support issues as both institutions and individual students struggle to gain and maintain network access. There is also less pressure for fundamental change in institutional processes or in teaching and learning.

Notebooks Provided by Program

This option may be easier politically to implement, particularly for students in technical fields who anticipate higher revenue streams once they secure employment or where there is obvious use of computers by professionals in the field. Passing along the cost of the notebook computer in this way is not conceptually different from higher differential tuition rates which may already be imposed on students in some professional programs.

Depending on how it is viewed within an institution, this model may limit possibilities for future institution wide initiatives. Or, it may serve as the "first wave" of implementation for an eventual campus wide commitment. Following this second line of reasoning, a campus might go with programs where there is the least initial resistance hoping that once others see the benefits additional programs would be added at later dates.

Desktops Provided to All Students

Providing standard desktop computers allows students and faculty to benefit from standardization and access to campus networks. Desktop computers typically cost less than notebook computers and capabilities are likely to be considerably greater. Institutions also find that costs of upgrades and peripherals are cheaper for desktops.

In spite of the initial appeal based on cost, a desktop model for implementation may not be cheaper when all costs are considered. Campuses must provide a large number of machines across many labs and in dorm rooms to meet faculty and student instructional needs. Like classrooms, many computers would need to be available for peak class times. At other periods of the day, computing resources might be idle. A mobile notebook model allows 24-hour universal access by directly handing the computer to the student and then expecting the student to carry the computer from one location to another.

Desktop computers are also fixed in their location. This may limit mobility for instructors or students who would like to move beyond the confines of the traditional lab setting. Service to the desktops requires that staff travel to the location where they are needed. This increases the resources needed for service and support. In a notebook model, the student or faculty member brings the computer to a central location. Northwest Missouri State developed a hybrid approach when their desktop machines came equipped with removable hard drives. This permits more flexibility in servicing hardware and minimizes the risk of lost data for students and faculty.

Notebooks Provided to Entering Freshmen

This model permits a campus to phase in a universal computer requirement over a period of four or more years. It is probably most popular with upper level students who question the degree to which faculty will use computers for instruction. Students nearing graduation may also initially resist a significant tuition increase to fund the technology project during their senior year. These students often recognize the benefits of the effort, but question the fairness of being required to participate at the end of their undergraduate career.

In spite of some initial appeal, the model does, however, carry some significant costs. Using this approach, the campus is pushed into a situation where it must designate a subset of multiple section courses as “notebook” classes. This may be easier on a larger campus with multiple sections, but would be difficult on a small campus where there are only single sections of most courses. If classes are not segregated, then faculty can not alter classes to take advantage of student use of notebooks. For a period of years, it is likely that some students will have the notebook computers and will not be participating in the initiative.

Additional problems arise for campuses where there is a new computer model selected every year. In this case, the campus is supporting at least four hardware/software combinations. There is increased complexity for both support and service. Faculty who work with students would not be able to assume common hardware/software capabilities among students they teach. This represents a greater problem in periods where consumers see significant enhancement of products by vendors.

Campuses using this implementation model must also continue maintaining traditional computer labs for a period of years. This intensifies service and support problems as the campus maintains the “old way” while it is also preparing for the “new way”. Another issue arises among some upper level students who may resent the focus of administrative attention and publicity on new students.

Notebooks for all Students

This model holds the promise of allowing campus leaders to open the window for significant changes in an institution. If the decision making and planning is handled well, the decision to distribute notebook computers across the campus can energize conversations about teaching and learning. All members of the campus community will be using the same hardware/software combination. This significantly eases support issues when any student or employee becomes a potential source of assistance. Campuses can also consider an immediate closure of traditional computer labs. Closing labs reduces demand on HelpDesk and computer center staff for lab support.

Other Issues

While it is not a distinction for the models offered in this paper, campuses differ in their approaches to providing service and support for student computers. Provision for an on-campus HelpDesk is the most common way in which institutions provide service, answer common questions, and check out machines for part-time student use. Some institutions identify a firm in the community as source for service. Similar distinctions arise when campuses consider the most efficient way to provide dial-in access to campus resources and the Internet. Many install and maintain modems for off-campus students. Others encourage students to make arrangements with a private Internet Service Provider (ISP). Some programs provide students with printers, while others centralize printing and encourage students to use the campus network with networked printers.

CONCLUSION

Faculty and staff on many campuses are working to improve student access to technology. Most of these initiatives rest on a belief that students will need better skills in use of computers. While each campus community necessarily pursues goals in ways which reflect unique cultures and missions, there are common models for creating technology intensive institutions which can be identified and compared.

SOURCES

Biros, Janice. (1998). Changes in Computer Purchase, Distribution, and Maintenance. Communications of the ACM, 41 (1), 43-44.

Brouwer, Peter. (1997) Information technology as a strategic resource: developing a lifecycle approach. Center Associate, AASCU Newsletter for Chief Academic Officers, 4 (3), 1-2.

Brown, David G., Burg, Jennifer J., and Dominick, Jay L. (1998) A Strategic Plan for Ubiquitous Laptop Computing. Communications of the ACM, 41 (1), 27-35.

Brown, Ray C. For an updated list of Notebook Universities with links to the respective campus web pages, see http://www.masu.nodak.edu/adminst/vpaa/thinkpadu98.html or http://www.vcsu.nodak.edu/offices/itc/notebooks/other.htm.

Brown, Ray C. (March 1997) Universal Access: Universities Benefit from Standardization. The Technology Colloquium, Microsoft Higher Education Newsletter. See also, http://www.microsoft.com/education/ hed/news/march/standard.htm.

Burg, Jennifer and Thomas, Stan J. (1998). Computers Across Campus. Communications of the ACM, 41 (1), 22-25.

Grier, Col. Samuel L. and Bryant, Lt. Col. Larry W. (1998) The Case for Desktops. Communications of the ACM, 41 (1), 70-71.

LeBlanc, Jr. Richard J. and Teal, Steven L. (1998) Hardware and Software Choices for Student Computer Initiatives. Communications of ACM, 41 (1), 64-69.

MacDougall, Glenn. (1998) Acadia University's "Sandbox". Communications of the ACM, 41 (1), 32-33.

McCandless, Glen. (1998) Creating a Level Playing Field for Campus Computing: Universal Access. Syllabus, 11 (6), 12-14, 29, 44.

McClure, Polley A., Smith, John W., and Sitko, Toby D. (1996). The Crisis in Information Technology Support: Has our Current Model Reached Its Limit? Boulder, Colorado: CAUSE.

Resmer, Mark, Mingle, James R., and Oblinger, Diana (1995). Computers for all Students: A Strategy for Universal Access to Information Resources. Denver: State Higher Education Executive Officers.

Rickman, Jon T. and Hubbard, Dean L. (1992). The Electronic Campus: A Case History of the First Comprehensive High-access Academic Computing Network at a Public University. Maryville, MO: Prescott Publishing Company.

Young, Jeffrey R. (December 5, 1997) Invasion of the Laptops: More Colleges Adopt Mandatory Computing Programs. The Chronicle of Higher Education, A33-A35. See also, http://chronicle.com/ colloquy/97/background.htm.

Using Faculty to Provide Technological Leadership and Support: A Critical Review

(Panel Discussion)

Mark K. Covey
Deborah Sullivan Trainor
Concordia College
Moorhead, Minnesota

As educational organizations move to implementing Information Technology (IT) advances into teaching, their core activity, many problems surface which require immediate responses. Issues such as bringing up networks, purchasing and installing hardware, and adopting software standards frequently are unanticipated consequences of the decision to become an IT campus. Yet these issues need to be effectively addressed if the IT implementation is to be a success.

Two substantial problems in this sea of challenges are technical training and support and technological leadership. By technological training and support, we mean those functions essential to using computers to accomplish routine tasks, such as word processing and electronic mail. In this category we include important subtasks such as formatting a floppy disk, loading paper into a printer, sharing files over the campus network, configuring a mailer, etc. While these tasks are now largely routine and mundane, those of us who are facile and comfortable with computers frequently forget just how counter-intuitive many of these routine tasks are to those of our colleagues with less experience in IT; yet surmounting these barriers is a very necessary first step in creating a college and university faculty that is technology literate.

By-in-large, these routine tasks can be taught by almost anyone with some background in computing. In fact, many faculty learn these tasks through the technological support available through their children. Others learn these tasks through painful experience or by consulting what technological support exists in the new computing environment. This venues to technology support seem to suggest that questions typical of the new user can be left to find their own answers in the academic environment; that is, these questions will “take care of themselves.” While this is true of many faculty, a significant portion of faculty will perceive this low level of formal support as a barrier to their adoption of technology. Additionally, skilled users who share their abilities with colleagues are quickly identified and overburdened with multiple requests for assistance. A prudent alternative then, is the provision of an appropriately educated technology support staff who can address these “low level” questions as they arise.

However three factors work against this model. First, technology support personnel in the current job market can demand premium salaries even if they have only minimal education or are new to their profession. The current job market is such that private business interests have been raiding college and university computer support staff to meet their own needs, frequently luring them away from institutions with lucrative compensation packages and incentives. Sadly, fiscal austerity is more a feature of institutions of higher learning than the corporate IT world. Academic organizations face a difficult choice between hiring fully capable personnel at budget-busting salaries or offering salaries which are unattractive to the very best support personnel available.

A second problem concerns status. Experience has shown that faculty members who have spent the better part of their young adulthood obtaining terminal degrees and establishing their professional identities through publications and high-level service often balk at taking instruction from individuals with less life experience or educational attainment. While we do not wish to unkindly characterize all faculty this way, a reality of higher education is that faculty spend much of their day giving instruction and direction to students; yet many faculty members are uncomfortable receiving instruction and direction, especially from individuals whom they do not regard as peers or colleagues. This problem is frequently exacerbated when faculty are taught by individuals who may be ineffective instructors because they are new to their professions, more concerned with technology's nuts and bolts, or frustrated by their "learned" pupil's inability to grasp what seems to be basic information and concepts.

A third issue concerns technological leadership. The very individuals who may be good at providing technological support are often removed from the classroom and the issues unique to that domain. Technological leadership in a teaching environment is often promoted by education practitioners yet technical support persons are seldom involved in teaching. Support personnel may be able to, for example, effectively instruct faculty in the procedures needed to connect a projection device to a laptop, but they may not be able to answer the more important question "why bother?" Faculty, especially those who have invested years in becoming effective lecturers and mentors, require clear rationales for modifying, or even abandoningpedagogies which served them effectively for many years.

The three problems of cost, status, and leadership may be effectively addressed by co-opting faculty as "technology consultants." By this model, a campus may appoint key faculty members with technology skills to serve as consultant to their colleagues. Cost issues are addressed because faculty can be given release time and their vacated responsibilities absorbed by the institution or replaced with temporary or adjunct faculty; certainly faculty release time costs less than an equivalent technologically-educated employee. Status concerns are answered as faculty learn from their colleagues and peers who are usually highly motivated to sustain inter-faculty professional relationships. Technological leadership concerns are addressed as the faculty technology consultants are also educational practitioners who are willing to take pedagogical risks and discuss the consequences of these decisions with their colleagues.

Ideally, the faculty technology consultant come from a variety of disciplines in the arts, sciences, and professions. Additional logical requirements include:

- some form of technological prowess; ideally, faculty technology consultants are comfortable with technology. They should have sufficient background in technology such that they can evaluate useful changes in IT from the hyperbole which often accompanies it;
- demonstrated technology leadership; faculty technology consultants should be among the first to use IT to *improve* their teaching in ways which enhance their existing pedagogies;
- willingness and ability to disseminate their experience to their colleagues; faculty technology consultants should be genuinely interested in sharing their discoveries with their colleagues and should have sufficient social skill to be able to do so.

During our panel discussion, we will describe, discuss, and evaluate Concordia College's efforts to provide technological support and technological leadership through the faculty technology consultant model. Funded by a grant provided by the Bush Foundation, the college appointed six faculty technology consultants who would lead and model IT applications in higher education. These faculty consultants conducted workshops for their colleagues, formed consulting relationships with individual academic departments, developed areas of IT expertise, and modeled IT pedagogies in their courses in an effort to increase the integration of IT into the college's courses.

Software Engineer Classes Develop an Outcome Assessment Software Tool

Janet M. Drake
University of Northern Iowa
Cedar Falls, IA 50614-0507
Email: drake@cs.uni.edu

Eric VanDaele
University of Northern Iowa
Cedar Falls, IA 50614-0507
Email: vandaele@cns.uni.edu

Abstract

This paper describes the development of a software tool for outcome assessment for a computer science curriculum. Program Assessment Tool (PAT) was developed by senior-level software engineering students over five semesters at the University of Northern Iowa. PAT is a survey tool that students use to show their understanding of topics taught in computer science classes. Unlike most student projects, PAT is a robust, usable tool. PAT provided a good learning experience because most students worked on PAT in a maintenance cycle and existing documents and code products made the software development goals more visible to the students. From the student point of view, we found that because so many students had worked on PAT, different parts were implemented using different functions of the implementation platform (MS Access). This made PAT hard to maintain.

Software Engineer Classes Develop an Outcome Assessment Software Tool

Janet M. Drake
University of Northern Iowa
Cedar Falls, IA 50614-0507
Email: drake@cs.uni.edu

Eric VanDaele
University of Northern Iowa
Cedar Falls, IA 50614-0507
Email: vandaele@cns.uni.edu

Introduction

A software tool was developed during five semesters of Software Engineering and Project Management classes in the Computer Science (CS) Department at the University of Northern Iowa. The tool, Program Assessment Tool (PAT), is for outcome assessment. Outcome assessment attempts to evaluate the effectiveness of the curriculum by examining how well students are prepared to perform in their field of study. PAT uses a student survey approach and students take a survey for each CS class at the end of the semester. The survey asks the students how they perceive their understanding of the topics taught in the class, relates the survey results to what the professor expects, and relates the topics in the survey to categories listed in the [ACM/IEEE 1991] report. In addition to the survey, PAT allows students to create records of the specific tools and techniques that they have used. This service is expected to be useful for students as a list of the tools and techniques they have used through out their education. The CS department is responsible for the maintenance of the system and individual instructors are responsible for making sure that the students in their classes answer the survey. PAT uses Microsoft's Access Relational Database System [Microsoft Corporation_1 1994] and is based on a Gateway 2000 DOS-based personal computer.

Outcomes Assessment

Like many other public educational institutions, the University of Northern Iowa has instituted an outcome assessment program. Outcome assessment aims at evaluating the students' and alumni's perceived results from the education they received. The question is:

> Did students get what they needed from their education to function successfully in their profession?

This is a difficult question and there are many approaches to answering the question. One approach is the "portfolio" where a collection of student materials are gathered throughout their program and examined at the end of their program. Another approach is to have students write a statement of goals at the beginning of their program and examine it at the end of their program. Alumni and student surveys are other approaches. At UNI we use several approaches. We survey student and alumni, the faculty estimates individual students writing, speaking, and teamwork skills, and students write a statement of goals when they enter the computer science program and this statement is reviewed when they graduate.

This paper describes how we developed the student survey approach and tool. Because we are a computer science department, we built a software tool, PAT, for student surveys. PAT was created by students over five semesters in Software Engineering and Project Management courses. The authors believe that PAT provided a good learning experience for students and a tool that the department can use. We are just starting to use PAT as a regular part of outcomes assessment and cannot yet comment on the ability of the survey approach to help with outcome assessment.

PAT, Survey Approach

PAT is a survey tool that surveys students after each of their CS courses. Students are asked about their understanding of topics taught in the course. They can select from 5 levels of understanding:

1. None: The student believes he or she has no understanding of the topic.
2. Low: The student recognizes the topic but has little understanding.
3. Medium: The student recognizes and has an understanding of the topic but has not used or implemented the idea.
4. Med-High: The student has an understanding and may have used the topic but does not feel mastery.
5. High: The student has mastered the topic and feels he or she can use the topic outside of the class.

PAT's survey is a blind survey because student names are not associated with their answers. Even though students use a password to enter PAT, the survey information has no relationship with student identification information. This protects students from the possibility of faculty punishment or personal embarrassment. In addition, students can only take one survey per class. This protects the survey from being skewed by disgruntled students or over-enthusiastic students.

The topics are selected and entered into the survey form by the instructor of the course. The topics are sentences or phrases up to 60 characters long and are grouped in four areas:

1. Abstract: Concepts and abstractions taught in the class.
2. Languages: Computer and design languages used in the class.
3. Skills: Specific skills and techniques taught in the class.
4. Tools: Computer tools used in the class.

In addition, students can add comments about the class to the survey.

The instructor also indicates the expected understanding level of the students. Not all topics introduced in a course are completely covered in the course. For example, the idea of execution time in big O notation can be introduced in the Introduction to Computing Course, added to in a Data Structures course, and covered in depth in an algorithms or theory course. The instructor is also asked to associate each topic with a topic in the [ACM/IEEE 1991].

PAT, Organization of the Tool

PAT is data centered system that has three main functional parts: Student, Professor, and Maintainer. The main goal of the tool is to optimize student survey function. Taking a survey must be quick and easy. The professor and maintainer functions are more difficult because setting up surveys, evaluating survey data, and maintaining current student lists and class lists are inherently more complex tasks.

The student enters PAT using a password. Based on the password, PAT displays a screen that list the surveys that the student can take. A student can take one survey for each class he or she is taking that semester. After a student has completed a survey of a class, the class will not appear on the entry list. [Figure 1] shows the welcome screen.

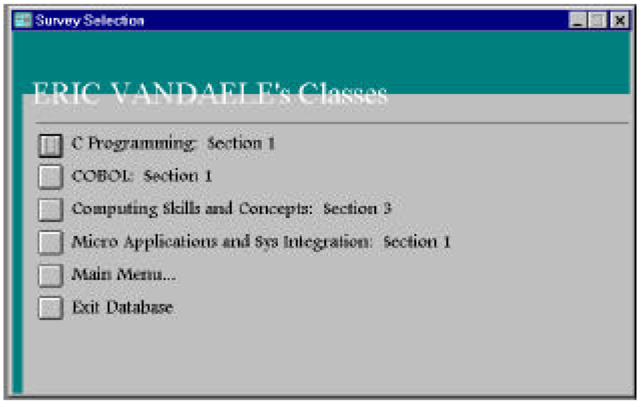
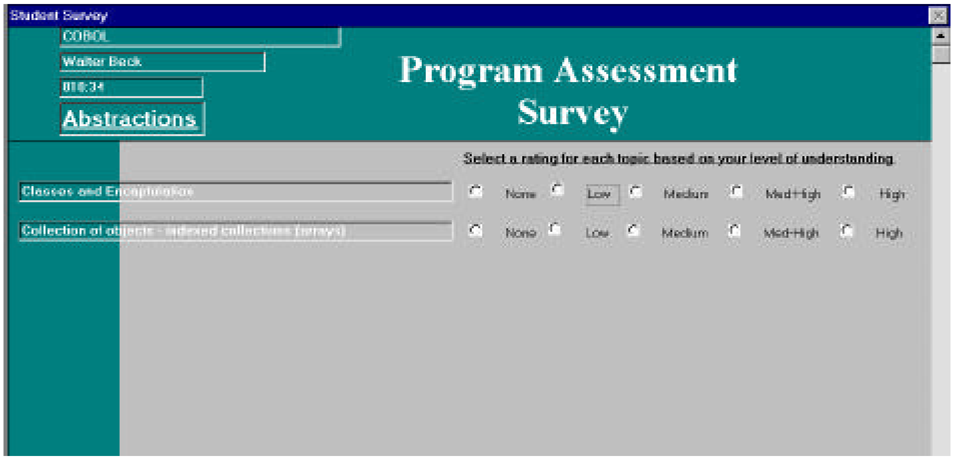


Figure 1: Student Class Choice.

When the student selects a survey, the survey screen appears [see Figure 2]. There are four survey screens, one associated with each topic group. The students can move between the four screens. The student can choose to complete the survey or leave the survey before completion. If the student does not complete the survey, the student is warned that no survey data is saved and that he or she must re-take the survey. We have used PAT in 11 courses and the average number of topic questions per survey is 30. One class with four laboratory sections took the survey in Fall 97. They took the survey in their last lab session. PAT was hosted only on one machine. Approximately 20 people completed the survey in two hours in each lab. That is approximately six minutes per student and none of the students had previously used PAT. We believe that our goal of ease of use was met.



Student Survey

COBOL

Walter Beck

010:34

Abstractions

Program Assessment Survey

Select a rating for each topic based on your level of understanding

Classes and Encapsulation	○ None	○ Low	○ Medium	○ Med-High	○ High
Collection of objects - indexed collections (arrays)	○ None	○ Low	○ Medium	○ Med-High	○ High

Figure 2: Survey Screen.

[Figure 3] shows the professor's task section and [Figure 4] shows the maintainer's task selection

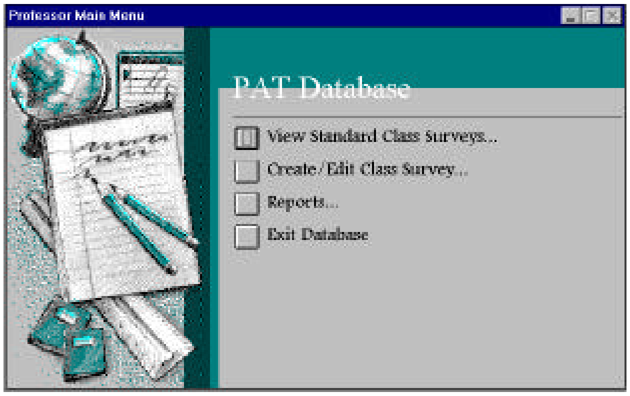


Figure 3: Professor's Task Selection

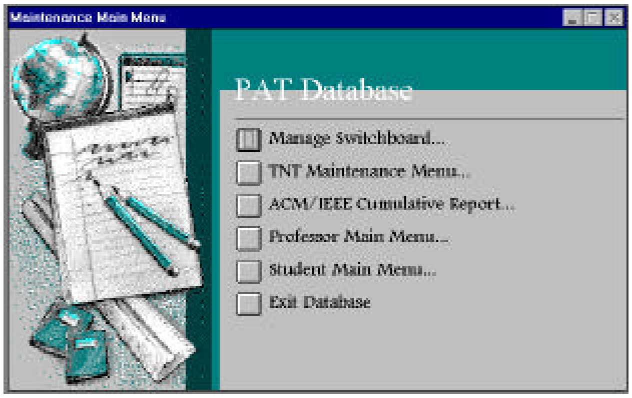
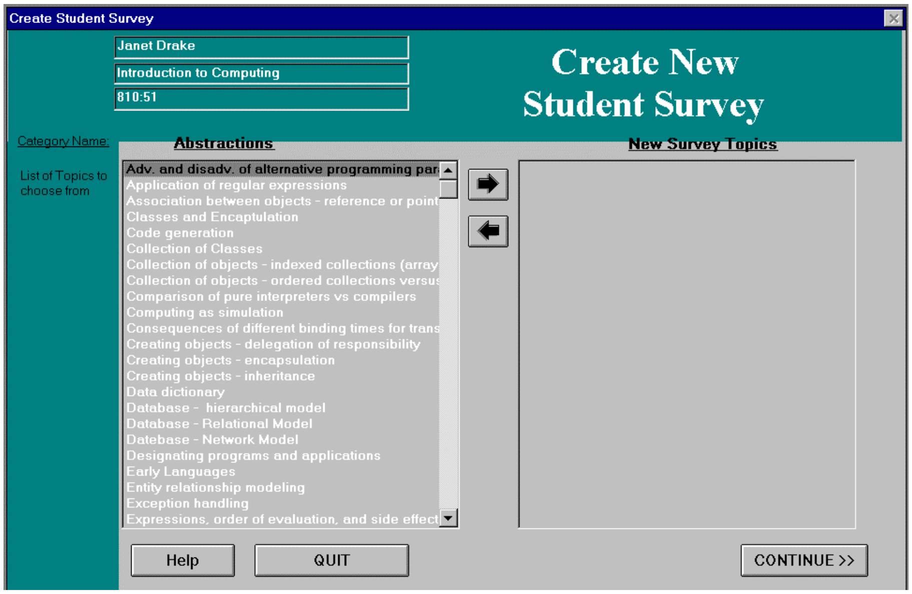


Figure 4: Maintainer's Task Selection

[Figure 5] shows the screen where the professor selects and adds topics to a class survey. After the professor enters all the topics, he or she indicates the expected level of understand for the students and associates each topic with a topic in the ACM/IEEE Curricula. The professor can also elect to make reports. Figure 6 is a portion of one of the reports.



Create Student Survey

Janet Drake

Introduction to Computing

810:51

Create New Student Survey

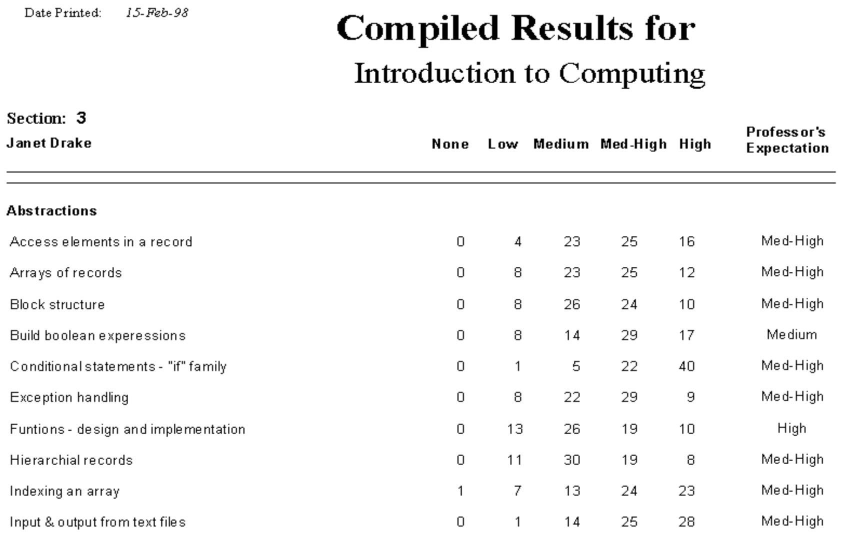
Category Name: Abstractions

List of Topics to choose from

New Survey Topics

Adv. and disadv. of alternative programming par
Application of regular expressions
Association between objects - reference or point
Classes and Encaptulation
Code generation
Collection of Classes
Collection of objects - indexed collections (array
Collection of objects - ordered collections versus
Comparison of pure interpreters vs compilers
Computing as simulation
Consequences of different binding times for trans
Creating objects - delegation of responsibility
Creating objects - encapsulation
Creating objects - inheritance
Data dictionary
Database - hierarchical model
Database - Relational Model
Datebase - Network Model
Designating programs and applications
Early Languages
Entity relationship modeling
Exception handling
Expressions, order of evaluation, and side effect

Figure 5: Professor adds and selects topics to a class survey



Date Printed: *15-Feb-98*

Compiled Results for

Introduction to Computing

Section: 3

Janet Drake

	None	Low	Medium	Med-High	High	Professor's Expectation
Abstractions						
Access elements in a record	0	4	23	25	16	Med-High
Arrays of records	0	8	23	25	12	Med-High
Block structure	0	8	26	24	10	Med-High
Build boolean experessions	0	8	14	29	17	Medium
Conditional statements - "if" family	0	1	5	22	40	Med-High
Exception handling	0	8	22	29	9	Med-High
Funtions - design and implementation	0	13	26	19	10	High
Hierarchial records	0	11	30	19	8	Med-High
Indexing an array	1	7	13	24	23	Med-High
Input & output from text files	0	1	14	25	28	Med-High

Figure 6: Report

PAT as a Teaching Experience – Instructors Viewpoint

The instructor learned the following lessons by using the PAT project:

1. Useful projects are more interesting to students. PAT is being used rather than just created as an exercise.

2. Maintenance cycles projects are more useful to students because they are the type of projects students are likely to work on in industry. PAT provided four classes with maintenance work.

3. Maintenance cycles are helpful to students because the project already has some projects such as a specification document, user manual, and code. It is easier to make something better than to start with a blank sheet of paper. PAT provided at least a specification and the prototype code to four classes.

4. Unjust criticism often crops up when students build on their predecessors' code.

5. Project maturity (prototype new idea, initial development, or maintenance cycle) effects the students' project experience. The following sections describe the different experience classes had working on PAT.

PAT was created over five semesters. The initial ideas for PAT came from the Spring '95 Project Management class. The Fall '95 Software Engineering class tested, proposed changes, and prototyped those changes. The Spring '96 Project Management class (all students had been in the previous class) implemented the changes. One more year (96/97) of maintenance in Software Engineering and implementation in Project Management left PAT in fairly good shape. During the summer of '97 the second author worked to polish PAT.

First class – Project Management

The assignment for the first class was to build a software tool for outcome assessment but no approach was specified. Three teams tackled the task. An expert from the Education Department came to class and talked about the problem. He ended by saying outcome assessment was impossible and discouraged us from trying. Despite the disheartenment, the teams continued working. One team selected a testing approach and two teams selected a survey approach. [Babbie 1989] describes designing surveys and provided guidelines for PAT's surveys. The survey teams also used the [ACM/IEEE 1991] as an external model to evaluate the curriculum coverage. The products include a specification document and a prototype tool from each team. The specification documents and the prototypes were weak. The students could demonstrate the prototypes only by following well-defined, safe paths. Even though the software was not good, the student teams had made great strides in 15 weeks. Coming from the assignment of outcome assessment, through an "impossible" prediction, to a sensible prototype was good work. The students did not feel particularly successful because they were not able to produce a complete product in 15 weeks. Although they knew the objective of a complete product in such a short time was not realistic, they still expected that they could do more.

Second class – Software Engineering

The instructor selected one of the survey tools for further work in the second class based mainly on the specification document. The assignment was to test the tool, propose changes, and implement several of the proposed changes. The class was critical of the prototype because it was incomplete. They somehow believed that they could have done better and failed to appreciate the challenges that the previous class faced.

Third class – Project Management

In the third class, sixteen students in four subgroups worked on PAT. They moved PAT from a Pascal implementation to a database implementation using MS Access. The class produced a set of fairly good products. The specification followed [MIL STD 498] and was greatly improved over the original by using software inspection. A user's manual was produced following the Story Board Approach [Weiss 1991]. Design and test documents were also produced but their quality was not as high as the other documents. The

PAT tool worked fairly well and UNI students were able to use it to evaluate their classes. The "Professor" and "Maintainer" functions were not as strong as the student functions. Overall the students felt quite successful. Because they did the initial design, implementation, and testing, they saw dramatic changes in PAT.

Fourth class – Software Engineering

Again the Software Engineering class tested, proposed changes, and prototyped several of the changes in the fourth class. This class was not as critical of their predecessors but still grumbled about coding style, interface features, and documentation.

Fifth class – Project Management

In the fifth class, eight people on two teams worked to clean up PAT. The students did not see the large improvement that the second class saw. The nature of the task had changed to maintenance. Changes were difficult to make and resulted in only small observable changes. The SRS was again improved and PAT was much more stable and usable.

Individual work – Single student's summer work

Specific maintenance tasks were selected for a summer project. The tasks mainly dealt with the "Professor" and "Maintenance" modules, and a more consistent use of MS Access features.

PAT as a Learning Experience – Student Viewpoint

The second author is the student who did the individual work on PAT. He had been in both the previous Software Engineering and Project Management classes and had a good understanding of the outcome assessment functions of PAT. On his teams in the classes, he took on the duty of learning MS Access and implementation. Because MS Access was not specifically taught in any CS course, learning MS Access without formal instruction was a major effort. Even after working on the maintenance project of PAT for six months, he found the individual tasks much more of a challenge than he had expected. Five major issues caused the maintenance task to be three times more time consuming than estimated:

1. MS Access is large and difficult to learn. PAT's developers learned to use MS Access on their own and found different approach to solving the same problem.
2. PAT has an event driven control model. PAT's developers did not implemented the control mechanism in a consistent way.
3. PAT had been implemented using a switchboard approach that proved difficult modify. The switchboard was a not the default approach and lacked some of the functionality of the default approach.
4. PAT's database had attributes and relationships that could be removed. PAT had been used enough that we knew what data an relationships were necessary and could eliminate others.
5. The project suffered catastrophic data loss from a tornado that destroyed the computer and all the backups.

The first issue was that MS Access is a large tool with a lot of functionality. PAT's developers learned the tool on their own. They learned different functionality and this resulting in different approaches being used throughout PAT. An MS Access application is composed of objects including tables, queries, reports, forms, modules, and macros. An application is made up of objects that the user sees and objects that control how the forms and reports work. Objects are built in the Design View and have a specific syntax that only applies to that certain form, report, and query. An example of syntax is that all [Microsoft Corporation_2 1994] form names must start with a capital letter in order for the database engine to recognize the form. All objects have certain properties to force them to behave and act in certain way. Properties are powerful and easy to use once the implementer understands the purpose of the property.

The second issue deals with the problem that the three different parts of the program used different approaches to implement events. Each program part was developed by a different team and the problem was not uncovered until PAT was maintained as a single tool. MS Access [Microsoft Corporation_1 1994] allows implementers to either use a macro or an event procedure to add the response to a certain event, which means that there are no standard ways of naming a macro or an event procedure. The problem comes into play when the macros and event names do not relate to the topic they are supposed to cover. For example, some of the macros where named macro1, macro34 and event procedures where given non-descriptive names. The process of figuring out what each event or macro did was difficult and time consuming. The three different parts of the program rarely used the same names for the same events. The lesson learned here was that naming conventions and data dictionary are necessary for maintenance.

The third issue deals with the implementation of the interface for the PAT program. A switchboard approach was initially selected to manage the all interfaces of the program. It worked well with the original seventeen inch monitor that PAT was implemented on. No screen re-sizing event was required until PAT was moved to a smaller screen. The switchboard coding approach did not support the re-sizing event of the forms or reports to a smaller screen. Inability to resize the screen was especially bad for the student surveys where the survey topic went off the bottom of the screen and students could not get to all the survey topics and, thus, the student could not finish the survey. We finally decided to replace the switchboard approach with the default approach. The lesson learned was that when using a new implementation platform, selecting the default approach is the best and easiest choice to maintain.

The fourth issue deals with the PAT database. The initial data model created in analysis was implemented and used in PAT. After using PAT for some time, we found that some of the many-to-many relationships were not necessary because the queries did not join over both directions of the relationships. The many-to-many relationships were expensive to maintain because join operations take time and complex trigger code is necessary to maintain the data integrity. We were able to collapse several many-to-many relationships into one-to-many relationships. In addition, we found several data fields within tables that were not used. We eliminated these fields to reduce the size of the database. The database changes were possible because we had used PAT for a long period of time and knew the necessary queries. The advantage from knowing the application well enough to modify the underlying database is a smaller database and time savings in data update and querying.

The last issue and a truly unusual learning experience is that mother nature plays a role in everything we do and we have no control over some events. A tornado destroyed the computer that hosted PAT and all the backup copies of PAT on a Friday night in June. The backups were next to the computer and the computer was in the car in a garage. All was destroyed. The computer had a six foot long 2 x 4 stuck through it and through the bottom of the car. This is something that I never want to hear or see again. All the maintenance was lost in the flash of a eye. This disaster taught me that rigorous backup and recovery is necessary even on a university project.

The major lesson learned was that even when the problem domain is well understood, maintenance activities take longer than expected. In this case, uncovering inconsistency in the event implementation approaches and the problem with control of the GUI cost more time than expected. We recommend that developers using unfamiliar implementation platforms select the default approaches or those taught in tutorials. We also suggest that rigorous configuration management and backup and recovery procedures be developed even for student projects. We do not expect catastrophic disasters with each project, but it is best to be prepared.

Conclusion

We developed an outcome assessment tool, PAT. We have used it in several of our courses and intend to use it in all our CS major's courses. Although students can easily use PAT and we can get summary reports on the surveys, we do not know exactly how the reports will effect our outcome assessment effort.

As a student project, we believe PAT provided a good experience because students worked on a real project rather than a throw-away project. PAT was a maintenance project for four out of the five semesters. We believe maintenance experiences are important because they provide a starting point – documents and code. Also, maintenance is more realistic because most working environments support or improve existing software rather than build totally new software.

PAT suffered from inconsistency in implementation because many student teams work independently. The implementation platform (MS Access) offered many implementation choices and teams elected different approaches. One suggestion is that the default approach or those approaches described in tutorials or through on-line tutorials be selected. Also, naming conventions should be used. Finally, configuration management and backup and recovery procedures should be instituted even on classroom projects.

References

[ACM/IEEE 1991] Report of the ACM/IEEE - CS Joint Curriculum Task Force (1991). Computer curricula 1991. IEEE Computer Society Press and ACM Press.

[Babbie 1989] Babbie, E. (1989). The practice of social research, fifth edition. Belmont, CA: Wadsworth Publishing Company.

[MIL STD 498] MIL STD 498 (5 December 1994). Military Standard, Software Development and Documentation. Department of Defense.

[Microsoft Corporation_1 1994] Microsoft Corporation. (1994). Building Applications: Microsoft access relational database management system for windows (Version 2.0). Microsoft Corporation: Author.

[Microsoft Corporation_2 1994] Microsoft Corporation. (1994). User's Guide: Microsoft access relational database management system for windows (Version 2.0). Microsoft Corporation: Author.

[Weiss 1991] Weiss, E. H. (1991). How to write usable user documentation. Phoenix, AZ: Oryx Press.

Teamwork in Computer Science Classes:
Advantages and Problems,
A Panel Discussion

Panel members:

Janet Drake -- Panel Chair	University of Northern Iowa, Cedar Falls, IA
Jim Schnepf	College of St. Benedict/St. John's University, Collegeville, MN
Karen Sutherland	University of Wisconsin-La Crosse, La Crosse, WI
Milton Wikstrom	Wartburg College, Waverly, IA

The traditional college class follows a competitive model where students compete for grades. If students help each other, they risk lowering their own grade by improving someone else's grade. The competitive model is the opposite of the model used in the software development industry – the team model. Teamwork is important in industry and often the difference between success and failure of a project can be traced to teamwork. Teams permeate industry and individual accomplishments are rare. Even the outstanding examples of individual success – Steve Jobs and Bill Gates – did their initial work with small, close knit teams.

Theory of teamwork exists. [Constantine 1993] describes four team styles:

1. The military model where a strong leader makes all the decisions.
2. The research model where the leader provides resources but little direction.
3. The no-leader model where all the team members know their roles and need no leadership or guidance.
4. The consensus model where all team members must agree on every decision.

Constantine says that the ideal software development team is close to the consensus model but includes a leader who provides resources and can make decisions when necessary.

[ACM/IEEE 1991] suggests teamwork in a computer science curriculum. It says team projects "add important breadth, depth, and realism to the curriculum." The panel members regularly use teamwork and describe their experiences with team projects in the classroom. Classroom teams have special problems compared to teams in industry. Classroom teams have leadership problems because they have no proven leader. The logistics of meeting times and meeting rooms are difficult for students. Also, not all students on a team want the same things out of a class and different motivations can cause contentions in teams.

The panel members will discuss their experiences with teamwork in the classroom. They are from different colleges and have used teamwork in different areas of the curriculum, from first courses to section courses. The issues covered include:

1. How to select teams.
2. How to evaluate teams.
3. What are reasonable expectations for team projects?
4. How to guide and support teams during a project.
5. Use of incremental assignment to keep teams motivated.
6. How to identify troubled teams.

[Constantine 1993] Constantine, Larry L., (1993). Work organizations: Paradigms for project management and organization. Communication of the ACM, 36 (10), 34-43.

[ACM/IEEE 1991] Report of the ACM/IEEE -- CS Joint Curriculum Task Force (1991). Computer curricula 1991. IEEE Computer Society Press and ACM Press.

Teamwork in Computer Science: Advantages and Problems Position Paper

Janet M. Drake
University of Northern Iowa
Cedar Falls, IA 50614-0507
Email: drake@cs.uni.edu

I believe that learning to work as a member of a team is an important skill for computer science students. They will work in teams in industry and should understand how people work together. I worked in industry before teaching and have worked on many teams: some good and some not so good. My organization believed in the importance of teams and taught courses in team skills and team building.

At the University of Northern Iowa, many of the faculty use team projects. Most students have done a team project by their junior year. I teach senior level software engineering courses and use teamwork in those courses. In the first software engineering course, students are in a team for eight weeks. In the second course, students are in a team for the whole semester.

I teach teamwork. The students read [Constantine 1993] that describes team theory. The paper describes four team paradigms and suggests a team paradigm for software development. The software development model is called a *structured open team* and is characterized by consensus and a leader who is first among equals. I also discuss the difference between industrial and student teams. Student teams have special problems with finding meeting times and meeting rooms. In addition, team members have different goals. Some members just want to pass the class while others want to be the best in the class. In addition, I use [Cleese 1993]'s movie, *Meetings, Bloody Meetings*. The movie is a practical guide to productive meetings.

Team Selection

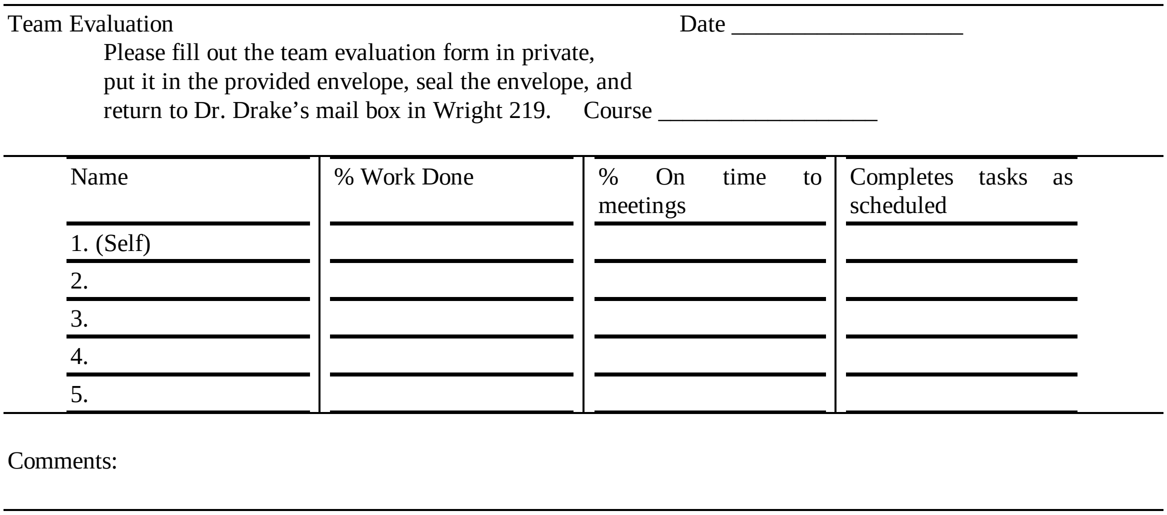
To select teams, the students complete a form that shows their free time and lets them request team members. They can also ask not to be on a team with someone. In the first team course less than half of the students request team members. By the last team course, most students will request team members. I try to honor the requests but do examine the students' free time to be sure that they can work together.

Team Evaluation

Team members evaluate each other. Students completes the form shown in Figure 1. The method for filling in and handing in the form tries to ensure privacy so that student will not suffer from social pressure. Team evaluations are used for at least 5% of the grade. If students behave badly in their team, I will reduce their final grade by at least one full letter grade. Most often the other team members will have spoken to me and I know the situation before the team evaluation. I have also found that team members will fill in for others when problems occur. I often get comments such as "Jane only did 15% of the work but she had health problems. I think she did all she could under the circumstances." The students are not penalized for helping each other because this is what teamwork is all about.

Identifying Troubled Teams

I estimate that about one out of six teams has some problem. The most common problem I see is where one student on the team does not want to work. Usually the other team members talk to me in private about the problem. They also use the team evaluation form. The other team members can usually work around the problem and the non-worker is socially ostracized and gets a grade penalty.



Team Evaluation Date ___________________

Please fill out the team evaluation form in private,
put it in the provided envelope, seal the envelope, and
return to Dr. Drake's mail box in Wright 219. Course __________________

Name	% Work Done	% On time to meetings	Completes tasks as scheduled
1. (Self)			
2.			
3.			
4.			
5.			

Comments:

Figure 1. Team evaluation form.

The second most common problem I see is when team members disagree and are unable to get any work done. They and I usually recognize the situation by the time the team gives their first presentation or delivers their first written assignment. To help the situation, I attend their team meetings, make sure everyone is assigned a task, and understands their task. Just my presence seems to be enough to make the team member reconsider their positions and come to some consensus.

The most difficult to recognize problem I have seen is the "nay-sayer". A team had one non-traditional student who already had a degree in mathematics and was considering a master's degree in computer science. He was negative – a nay-sayer – on every assignment and affected the other team members. By the end of the semester the other team members and I came to realize that the non-traditional student had a negative effect on the team's performance. The whole team suffered because we were unable to identify the situation early in the semester.

Another bad team situation was when two students who considered themselves to be superior did not allow the two others on their team to do any work. They worked late at night and did not inform the others. They gave their fellow team workers bad evaluations but did not consider that their attitude and behavior had anything to do with the low percentage of work the others did.

Incremental Assignments

People get work done when there is a deadline or due date. I saw the same pattern in industry. People work harder when the deadline is near. Thus, I assign teams work at regular intervals. Intermediate products such as specification documents, design documents, test documents, and prototypes are a regular part of software development and provide good intermediate products.

Team Advantages

As a whole, I believe teamwork is beneficial to students. They learn to work with others and to appreciate the talents of others. I have seen teams find that they have a member who can write or have has artistic talent. Many students enjoy their team experience. About one third of the teams comment on the evaluation form that their team was great and they enjoyed the experience. I believe that more students have an excellent team experience than those who have some trouble on their team. Even teams that have problems such as one lazy member still manage to enjoy the experience.

Another big advantage is that teams help students learn difficult concepts. I teach Entity Relationship Modeling and Structured Analysis. Both have many difficult concepts. The first assignment is an individual assignment and many students do poorly and obviously do not understand. They repeat the assignment as a team. Working together they usually master the concepts and the quality of the product is greatly improved.

Conclusion

I believe teamwork is necessary in college courses but not all courses and not all assignments should be team assignments. I believe team theory should be taught so the students understand team models and know what to expect from their team in college and in industry. Team work requires that the professor be sensitive to grouping people together and watching the teams to make sure they are progressing properly. I also believe that the professor gives up tight control over the class when students work in teams because he or she cannot track the work of individual students.

References

[Cleese 1993] Cleese, J. (1993). Meetings, bloody meetings. Video Arts Ltd Chicago, IL.

[Constantine 1993] Constantine, Larry L., (1993). Work organizations: Paradigms for project management and organization. Communication of the ACM, 36 (10), 34-43.

Teamwork in Computer Science Classes
A Panel Discussion

When the Teams are Separated by a Thousand Miles

Karen T. Sutherland
Department of Computer Science
University of Wisconsin - La Crosse
La Crosse WI 54601
suther@csfac.uwlax.edu

The teamwork in the classroom experience took a unique twist on the University of Wisconsin - La Crosse (UWL) campus in the Fall of 1997. Three member teams in our upper level Artificial Intelligence class were teamed with two member teams from an upper level robotics class offered in the Department of Mechanical Engineering at Wilkes University in Wilkes Barre PA. The students not only had major differences in philosophies, backgrounds and career goals, but were widely separated by physical distance as well.

The group project was to control a Puma robot arm at Wilkes through a user interface designed at UWL. The task for each group was identical: to pick up a randomly placed set of colored objects from a table and place them in boxes based on color. The Wilkes students designed and built robot end effectors for the task. The UWL students designed heuristics to do as much of the task as possible autonomously (e.g., identify a block by color, move to the block, grasp it, move to the box and drop the block). They were allowed to design low level macros to accomplish tasks such as moving to a given box.

The projects were demonstrated to the instructors at the end of the course. In addition, each group was required to turn in project plans, periodic progress reports, and a final formal report detailing the development of their project. They also kept and turned in a log file consisting of all email messages both between the group members themselves and between the group members and faculty.

Groups were self chosen by the students on the individual campuses and were arbitrarily matched on the two campuses.

What we learned:

1. This type of cooperation requires the instructors to constantly remain on top of what is going on in every group. Tasks must be clearly specified at the beginning of the project and changes should be avoided.

2. Communication between group members, campuses and instructors is critical. What the students thought was clear communication was often misinterpreted. Internet ``talk'' sessions were useful, but when anything important was decided, the communication was via email. In this way, a written record was always available. This log file was a great help in ironing out several ``But, you said...'' type arguments. In fact, since there was never a question of what anyone said, arguments of this type ended more quickly than in the more traditional group setting.

3. Students from the same programs have different priorities and goals. We found that students from different programs can have significantly different (and conflicting) priorities and goals.

4. Faculty members in different programs also have different priorities and goals. In this case, one was looking mainly at the job that was done while the other was concerned with the skills that the students were learning. What some might refer to as the engineering point of view vs. the AI point of view was apparent. The engineering faculty maintained that if everything was not picked up and put into the correct boxes, the group would fail. On the other hand, keeping in mind that any good heuristic works well most of the time but is not guaranteed to work all of the time, the AI faculty refused to consider a dropped block a failure.

5. The usual team styles (strong leader, no leader, consensus, etc.) were complicated by the fact that there were essentially two sub teams. When teams had two strong leaders, one on each campus, problems occurred. The lack of physical proximity played a part in keeping a single leader from emerging. Likewise, it was difficult to get everyone to agree when they were not meeting face to face, making the consensus model ineffective.

6. Due to the nature of the project, there was no worry about finding meeting rooms. The students worked in their respective labs. However, the time constraints involved all group members plus the hardware. Only one group could use the manipulator at the same time. This sharing of resources combined with the fact that someone had to be at each end for anything to be accomplished, complicated the issue.

7. This was a good exercise in group dynamics. It clearly identified group members who were good communicators. However, improving communications skills was not a priority goal of the project.

8. It appeared that it was easy to blame someone who wasn't present, which led to more blaming than either instructor had experienced with traditional groups. This, of course, could have been due to the particular individuals involved.

Would we do it again?

Certainly. Any type of substantial project in either of these fields cannot be done by a single individual. The project addressed here could not easily be done by individuals in either group alone. Computer scientists do not usually know much about building end effectors and mechanical engineers are not traditionally skilled in AI programming techniques. We do, however, believe that the major benefit in doing this type of project lies in the experience of doing the work itself, not in the group experience.

The Group Semester Project - The Fundamental Task

Dr. Milton C. Wikstrom
Department of Math, Computer Science, and Physics
Wartburg College
Waverly, IA 50677-1003
wikstrom@wartburg.edu

Abstract

One of the most effective teaching tools that I use in the CS1 and CS2 courses is the use of team projects. It is my belief that the benefits of team based projects far out way the negatives. In this position statement, I clarify these beliefs and state my reasons.

The Group Semester Project - The Fundamental Task

Dr. Milton C. Wikstrom
Department of Math, Computer Science, and Physics
Wartburg College
Waverly, IA 50677-1003
wikstrom@wartburg.edu

Abstract

One of the most effective teaching tools that I use in the CS1 and CS2 courses is the use of team projects. It is my belief that the benefits of team based projects far out way the negatives. In this position statement, I clarify these beliefs and state my reasons.

Position Statement

It is my firm belief that heavy use of team projects in programming courses is an absolute requirement for the effective teaching and learning of the programming process. While I recognize that there are several drawbacks to this approach, I believe the benefits outweigh them. In the rest of this statement, I will first discuss the negative aspects. Positive aspects are then discussed.

Negatives of Team Projects

The authors of [Coleman and Thibault 1995] point out the most commonly heard drawback to the use of team projects. Quoting from their paper, they said, "The pitfalls include the overhead of team communication and coordination, and the potential for some team members to not contribute their fair share of the work on the project."

Furthermore, a potential disaster can easily occur within a group that takes on too large or too unfocused a project. This was the main concern expressed in [Bareiss 1996]. Groups that wait too long to get started are also at risk. Ultimately, this implies that an instructor must either keep a careful eye on all groups or leave them to their fate.

Hence, instructor workload is another concern. It is a fact that group and project management *can* eat up a lot of time.

Benefits of Team Projects

Next I examine some counter arguments to the above points. In the previous section, [Coleman and Thibault 1995] correctly express concerns regarding team communication and work distribution. However, they then propose a peer evaluation instrument as a solution to this problem. In [Wikstrom 1997], I stated that their solution was effective but was overkill. My own experience shows that other grading instruments, such as exams, act both as motivators and equalizers to slackers.

Moreover, in [Coleman and Thibault 1995], the authors support the group project concept by stating: “College students need to experience group dynamics and to learn the benefits and pitfalls of working in a team environment. The benefits include division of labor and alternate viewpoints on how to complete a project.” Similar findings are found in [Briggs 1991] and [Bareiss 1996]. [Bareiss 1996] also states that the project concept gave the students something they could be “proud of.”

In [Beidler et al. 1985] it is stated that for CSIS 1 & 2, “The students receive an overview of programming methodology and learn to write programs following good style and accepted practices.” My personal experience is that students don’t *really* learn how to program until they have attempted a major project of their own. In my classes, the final project must be presented to the class. Hence, the foreknowledge that the code will be shown to and judged by their peers has a very positive effect on style, quality, and efficiency issues.

From the instructor’s perspective, there is the additional advantage of having “fewer submissions to mark” and for “allowing more variety in the types and complexity of assignments [Becker 1994].”

Finally, the single most obvious reason to use team projects is that this most closely models the way software is written in industry. I realize this is a bit of a sacred cow in academia, but having worked in industry, I believe it to be the most efficient model.

References

[Bareiss 1996] Bareiss, Catherine C., “A Semester Project for CS1,” *Proc. 27th SIGCSE Technical Symposium on Computer Science Education*, (Feb. 15-18, 1996), 310-314

[Becker 1994] Becker, Byron Weber, “Inexpensive Teaching Techniques with Rich Rewards,” *SIGCSE Bulletin 26,4* (Dec. 1994), 41-44

[Beidler et al. 1985] John Beidler, Richard H. Austing, and Lillian N. Cassel, “Computing Programs in Small College,” *Communications of the ACM* 28,6 (June 1985), 605-611

[Briggs 1991] Briggs, Jim, “Group Projects in Software Engineering at York,” *SIGCSE Bulletin 23,4* (Dec. 1991), 48-50

[Coleman & Thibault 1995] Coleman, David L. and Thibault, Henry C., “Peer Evaluations for Team Projects in Computer Science Courses,” *Proc. 28th Annual Small College Computing Symposium*, (April 21-22, 1995), 34-42

[Howatt 1994] Howatt, James W., “On Criteria for Grading Student Programs,” *SIGCSE Bulletin 26,3* (Sep. 1994), 3-7

[Wikstrom 1997] Wikstrom, Milton C., "Group Semester Projects in CS1 and CS2," *The 30th Annual Small College Computing Symposium*, University of Wisconsin - Parkside, April 1997

Solving the Puzzle of Matching Content to Technology

Betsy Draper
Instructional Technologies Center
South Dakota State University
USA
draperb@ur.sdstate.edu

Denise Peterson
Instructional Technologies Center
South Dakota State University
USA
petersd@ur.sdstate.edu

Madeleine Rose, PhD, RD, LN
Nutrition and Food Science Department
South Dakota State University
USA
rosem@itctel.com

How do you deliver a course that has low enrollments and is taken by graduate students located across the state? What looked like a situation for disaster turned into a valuable opportunity to promote what all of us in the educational technology field dream about ... focus on content, then apply the appropriate technology to create a student-centered learning environment. The goal of effective and efficient delivery is to reach the target audience, matching the content with appropriate technology at a reasonable cost. This decision is based on various factors including content, location of students, technology skills of the students, technology accessibility, time, and faculty input. Putting this course together required a decision-making process, design, development, implementation, training and evaluations. The authors also produced a "report card" system of evaluating technologies for various situations.

Solving the Puzzle of Matching Content to Technology

Betsy Draper
Instructional Technologies Center
South Dakota State University
USA
draperb@ur.sdstate.edu

Denise Peterson
Instructional Technologies Center
South Dakota State University
USA
petersd@ur.sdstate.edu

Madeleine Rose, PhD, RD, LN
Nutrition and Food Science Department
South Dakota State University
USA
rosem@itctel.com

Introduction

How do you deliver a course that has low enrollments and is taken by graduate students located across the state? What looked like a situation for disaster turned into a valuable opportunity to promote what all of us in the educational technology field dream about ... focus on content, then apply the appropriate technology to create a student-centered learning environment. Classes using one type of technology for delivery are now commonplace. However, technology delivery has a wide variety of costs, depending on the type of technology used. The goal of effective and efficient delivery is to reach the target audience, matching the content with appropriate technology at a reasonable cost. This decision is based on various factors including content, location of students, technology skills of the students, technology accessibility, time, and faculty input.

This is a case study of a course, offered through a public four-year institution, that used several different technologies to best meet the instructor's course objectives. Additionally, the course was designed and adjusted as students' skills changed. We will discuss the decision-making process, design, development, implementation, training and evaluations we incurred while putting this course together. From this work, the authors also produced a "report card" system of evaluating technologies for various situations.

South Dakota State University (SDSU) has a long history of delivering courses through interactive video. According to University documents, in 1989 the Vice President of Academic Affairs established a task force on telecommunications. In the Fall of 1990, via an educational television microwave link, the first interactive video course in the state was offered. Training was provided for faculty in the Summer of 1991 and in the Fall of 1991, four courses were offered using this system. The state of South Dakota formally established the Rural Development Telecommunications Network (RDTN) in 1992 to provide two-way video and audio distance learning.

In the spring of 1996, the state's governing board established technology in the curriculum as a priority by creating the "Redirection for Efficiencies" program. SDSU had already supplemented their interactive video facilities with two additional classrooms. Ten smart classrooms (with projection system, network connection, document camera, video tape player, and laser disc player) were in operation in the spring of 1997. In the Fall of 1997, an additional interactive classroom, nine smart classrooms and the "Governor's Electronic" classroom (with ISDN capabilities and networked student workstations) were completed.

SDSU has also received a Title III grant to increase the literacy and access to computers for faculty. The funds have provided laptop computers to faculty so they could use computer technology to aid in more effective teaching. According to the SDSU's 1997 summary report, 144 instructors have completed computer training.

In the summer of 1996, Dr. Madeleine Rose decided that PowerPoint[1] presentations were the way to more effectively get her point across in large (50 - 120 students) classes. She had seen document camera demonstrations in a faculty development workshop at SDSU and decided that would work well to present some of her course materials. She shared a departmental laptop until she was accepted into the Title III training program in spring 1997 and received her own Toshiba laptop.

NFS (Nutrition and Food Science) 660, Maternal and Infant Nutrition, was offered in the summer 1996 session. Typically, the students interested in the course are employed as dietitians, nutritionists, or nurses in the pediatrics or obstetrics/gynecology fields. Although there was significant interest in the course, registrations did not meet the seven-student minimum requirement. The most viable solution was to bring the class to the students. In the summer of 1997, NFS 660 was offered again, but this time using alternative delivery. The students enrolled in the course, knowing they could maintain their current employment status and not have to spend their summer in Brookings.

Planning

Decision Making Process

This course represented the first at SDSU to be offered using a mixture of technologies. Planning was extremely important to ensure that every detail was worked out ahead of time and alternatives were available if one of the delivery methods failed. Another important aspect of planning was creating a team of support staff to assist the instructor. The team that worked with Rose consisted of Betsy Draper, Instructional Designer, and Denise Peterson, Distance Education Specialist. Other staff were added as needed to provide specific expertise, such as training and technical support. Draper was heavily involved in the planning process. Peterson, who arrived on campus just one week before the course began, worked closely with Rose during the actual delivery of the course.

The Department Head, Rose, the Director of Instructional Technologies, and Draper, met seven weeks before the course started. They discussed the possibilities of offering the course using means other than RDTN. The Department Head was new in the position and new to SDSU. She was very interested in trying innovative techniques. Draper was also new in the position, and this course offered a means to quickly learn what technologies were available at SDSU. After the initial meeting, the Department Head and the Director of Instructional Technologies decided that Rose and Draper should continue to investigate alternatives to RDTN-only course delivery. The Department Head shared concern that the departmental teaching budget would not support anything too extravagant. She and the Director of Instructional Technologies discussed meeting with the Associate Vice President for Academic Affairs and the Dean of the Graduate School to discuss additional resources and ensure that the course would meet any delivery requirements of the graduate school.

Later that same day, Rose and Draper met to do some brainstorming. Rose was willing to try some new things. She was obviously confident about her ability to teach the course content, additionally she had

[1] PowerPoint is a product of Microsoft.™

experience using RDTN, working with PowerPoint to develop presentations, and developing web pages. Although discussion initially focused on a replacement delivery method for RDTN, discussion shifted to using alternative technologies depending on the content and instructional delivery needed. Rose planned to use a series of lectures, specialized articles, and guest speakers to present course materials. Additionally, she planned to use discussion, writing assignments, and have each student develop a presentation that would be presented to classmates and a workshop that was open to the public. After brainstorming on potential technologies that could be used, the following list was developed:

- RDTN
- Web pages
- PowerPoint
- Email
- On-line chat rooms
- Audio conference (telephone)

Since the course was offered during the summer session, scheduling RDTN time was less difficult than during the academic year. Yet it was very important to get on the RDTN schedule as soon as possible to reserve locations and time slots. Rose reserved RDTN time for each contact hour of the course. This provided the first level of backup. If other technologies failed, RDTN could be available.

Over the next three weeks, prototype web pages were developed, on-line chat facilities tested, and PowerPoint presentations uploaded and download. The plan was developed enough so that a proposal could be presented to the Associate Vice President for Academic Affairs and the Dean of the Graduate School. They were open to the idea of multiple delivery methods, but wanted to see some budget figures. Rose and Draper were assigned to put together a budget estimate for the Department Head and the Director of Instructional Technologies.

Now it was time to add detail to the plan. Campus computing facilities could easily handle the web pages and email, but had no on-line chat facility that would work well across the Internet. Alternative on-line chat systems, available from other Internet sites, were evaluated. Recognizing that using a public chat room could potentially leave the students vulnerable from the aspect of connectivity and/or privacy, consideration was given to other technologies that would provide a “live” audio connection between all students in the class. Consideration was also given for students who might not have higher speed Internet connections to support audio over the Internet. Additionally, there would be the need for a microphone and speaker system for each student. A backup for on-line chat was selected -- an established technology, the telephone. Draper made contacts with a system to handle conference calls. Then she identified the procedures and timetables required to make the necessary reservations.

A spreadsheet was developed that listed delivery costs depending on technology used, number of sites, and number of students. With three weeks remaining until the course started, Rose and Draper met again with the Department Head and the Director of Instructional Technologies, reviewed the spreadsheet and options and decided to move forward.

Throughout this planning process, Rose was busy recruiting students for the class. The marketing approach was to let potential students know that using a different approach in delivery, they would spend less time on the road driving to a delivery site. Interest developed from the Rapid City area, Sioux Falls, Eagle Butte, and Sisseton. Due to cost, we were limited to the number of RDTN delivery sites. Brookings, Sioux Falls, and Rapid City were selected. Eagle Butte and Sisseton students would have a 180-200 mile round trip to the nearest RDTN site.

Rose and Draper worked together developing web pages for the class. These pages would serve mostly an administrative role, providing general course information, such as assignments, due dates, and grades. Using templates developed by Draper, Rose completed the course web pages in two days. Additionally, students would have individual web pages linked to the course page so they could get to know one another better, and upload their presentations to share with their classmates.

Following some lengthy discussion, Rose and Draper decided that student success would depend partially on what type of Internet access and experience the students had. Additionally, they needed to know the students' general computing skills and presentation software experience (e.g., PowerPoint). To adequately support the students during the course, they needed to be at a certain technology skill level. Rose developed a survey (see Appendix A) to be completed by each student to assess their technology experiences and skill level. Rose contacted interested students to find out if they could come to campus over a weekend for an orientation and training session.

Rose began lining up her guest speakers, letting them know that they would be presenting over RDTN or via a teleconference. She put together a syllabus that included a grid of what technology would be used and the duration of the use. Using the syllabus draft, Draper plugged numbers into the budget spreadsheet and came up with the numbers to provide a more up-to-date estimate of the delivery costs. Once the syllabus was finalized, reservations were made for teleconference sessions in the event that on-line chat facilities did not work out. Backup reservations on RDTN for these sessions were canceled.

The orientation and training session was organized. Presenters and trainers were scheduled. Two weeks before the course started, Rose sent the students the skills survey and included a cover letter with the orientation and training session schedule. Draper and Rose reviewed the returned surveys and made adjustments to the orientation and training session schedule. Noting that one student had significantly less experience than others, Draper contacted the remote site closest to that student to request support for some of the course activities. The staff at the site were very cooperative.

Orientation and Training Session

Based on the results of the skill survey, the orientation and training session consisted of 19 contact hours divided into 8 sections:

- Face to face instruction with Rose
- Group discussion
- World Wide Web training (including searching)
- Hypertext Markup Language (HTML) file creation
- Audio conference
- RDTN
- PowerPoint
- Guest speaker over RDTN

To accommodate student schedules, two two-day orientation sessions [Tab. 1] were scheduled.

	Location	Topic	Instructor
DAY 1			
8-10am	NFA 440	Pregnancy	M. Rose
10:00-10:30am		Break	
10:30-12:00pm	NFA 440	Pregnancy; Discussion of Abstract	M. Rose
12:00-1:00pm		Lunch	
1:00-2:50pm		Iron Deficiency Anemia Case Study	M. Rose
2:00-3:00pm	Library	First Search; evaluating web sites	M. Kraljic
3:00-3:30pm		Break	
3:30-5:00pm	NFA 440	Lactation	M. Rose

5:00-5:30pm		Break	
5:30-6:30pm	NFA 440	Lactation	M. Rose
DAY 2			
8-10am	PC 214	Picture RDTN (Lactation Abstr); Phone Bridge	B. Draper D. Peterson
10:00-10:30am		Break	
10:30-12:30pm	NFA 138	Netscape; Biography HTML; upload / download; email; chat (Lactation Abst)	B. Draper M. Rose
12:30-1:30pm		Lunch	
1:30-2:30pm	NFA 440	Critically Evaluating Research; Specker Article	M. Swanson
2:30-3:00pm		Break	
3:00-5:00pm	ADM 103	Power Point	J. Ullery

Table 1: On-Campus Orientation and Training Schedule

RDTN

As mentioned previously, the class would use the state's interactive video system for some of their class meetings (see Appendix B). The RDTN system is a two-way video and audio interactive classroom equipped with monitors, cameras, microphones, an overhead camera, and, at most sites, computer capabilities. An orientation was especially important for two reasons, 1) most of the students had not previously participated in an RDTN course or conference, and 2) students would give their final graded presentation over RDTN.

The students were given an orientation by Denise Peterson, the Distance Education Specialist, and Rose. Two classrooms were connected to simulate an RDTN course. Rose was in SDSU I and the students and Peterson were located in SDSU III. The students watched a five minute introduction tape developed by SDSU and then Peterson discussed some procedures to follow in class. Rose and Peterson then proceeded to demonstrate the abilities of the classroom including PowerPoint, Internet and chroma keys. Presentation tips were discussed and students were encouraged to play with equipment and view the control room. Rose then proceeded to present a lecture "via distance" by presenting in one classroom to the students in the other classroom.

Audio Conferencing

Time was spent on a conference call by connecting the students in one room using a conference phone to the a telephone in another room where Rose was located. Rose then presented an actual class "via distance." The PowerPoint presentation was operated by a student in studio III so that the students could use that technology while they were using audio conferencing. As Rose spoke, the student advanced the PowerPoint presentation.

Computer

Another piece of the orientation included computer training using actual course materials. The students received PowerPoint training so they would be able to put together a PowerPoint presentation for a class project. Students also received instruction on how to use Netscape[2] to browse the World Wide Web, and search the Web and evaluate information found on web sites. The students had fun with on-line chat facilities, but struggled with technical difficulties.

[2] Netscape is a product of Netscape Communications, Inc.™

Because the instructor was requiring students to use email and view PowerPoint presentations over the web, time was spent hands-on with email and uploading and downloading PowerPoint presentations. The students also created biographies in this orientation period. Draper had taken their pictures using a digital camera and loaded them onto the SDSU web server. Students then were shown how to create a biography using HTML templates provided by Draper and linking their picture. The biographies provided an opportunity for the students to get to know one another during the class.

The rest of the orientation period was used by the instructor to review materials with students and create a relationship to base future interactions upon. Since the students had to work together to create the final presentation, the social interaction and establishment of relationships were important to their success as a group. By the end of this orientation period, students appeared to understand the technologies they were using and had contacts to help them along the way.

Class Begins

The plan was developed around the instructor’s content. The choice to require students to be on campus for the orientation was for their benefit as their skills varied. The rest of the class was then scheduled using a combination of technologies. The Rural Development Telecommunications Network (RDTN) is an interactive video system statewide. Students were sent to three sites to attend the class using RDTN - Brookings, Sioux Falls, and Rapid City. The phone conferencing allowed some flexibility although it became evident that the students got together to participate this way as well. The chat room was dropped as there was some difficulty in using this technology.

RDTN

The Rural Development Telecommunications Network was used for 10 hours of class time. Three sites were involved in the class: Brookings where the instructor with three students were located, Rapid City with two students and Sioux Falls with three students. Each time the class was held over RDTN, there was a control technician available in Brookings. Both the Sioux Falls and Rapid City sites ran from a student controlled AMX. Peterson, the Distance Education Specialist, made arrangements for the site coordinators at the Sioux Falls and Rapid City sites to be available the first night of class to show the students how to run the AMX. Also it was important the students knew who to contact if there were any problems. Twice RDTN was used for two hours per night with 1 hour of phone conferencing to finish the night. Twice the RDTN system was used for a full three hours for class.

Overall, RDTN worked smoothly for the class. There was only one night toward the end of class where a remote site experienced tiling. The instructor used the system and control room technician to the fullest. The instructor used guest lectures, videos, PowerPoint and Internet to enhance the students learning opportunities and for the most part students seem to like the format. Finally, the students gave their final presentations using the RDTN system. There was interaction among all three sites as well as a team taught lesson from one individual in Sioux Falls and another in Brookings. From all three sites, they presented an informative public presentation for expectant mothers.

Audio Conferencing

The class used audio conferencing for 11 hours of the class. A three-way conference call system was set up. At SDSU, a room down the hall from the RDTN classroom was set up with a conference phone. The conference calls often took place after two hours of RDTN. Three nights of class were conducted for three hours over the conference phone. The phone at SDSU allowed for multiple speakers on this end with the ability to mute. The instructor placed the calls to Sioux Falls and Rapid City. At Rapid City, the site facilitator made a speaker phone available in the RDTN classroom for the students. The connection worked well. The site in Sioux Falls did not connect as easily. No phone or room near the RDTN classroom was available to the students. The NFS department was prepared to donate a speaker phone to those students, but a phone became available. Fortunately, one student was employed in the building and had access to conference rooms when the RDTN classroom was busy. However, the calls were rarely made to the same phone number. And on a couple of occasions the students had to move twice in one night.

The instructor used the audio conferencing for presenting materials, guest lectures and student presentations. Several students gave their presentations using audio conferencing. As mentioned before, students emailed their PowerPoint presentations to the instructor who then loaded them into the class web page. The students then downloaded the presentations on their own. Then the student presented during the audio conference as the other students viewed the PowerPoint presentation at their sites.

Internet

The instructor decided to use Internet technology to communicate with her students and for student to student interactions. She used a class web page, email, and PowerPoint to facilitate the learning opportunities for her students. The technology was not reflected in the actual class schedule. On-line chat was dropped and replaced by audio conferencing. The students had struggled with the on-line chat during the orientation session and Rose did not want to risk problems for the students. However, during training it was stressed to students that they should check the syllabus on a regular basis to follow the changes and announcements the instructor made. The instructor also used the syllabus to provide links to Internet resources that were related to the course and links to course project grades.

Results

Feedback (see Appendix C) from the students clearly indicated that they liked having a mix of delivery methods. The only negative feedback received was related to the length of a session on the audio conferencing and RDTN technologies. The class met two evenings a week for three hours at a time. One student respondent commented about having difficulty connecting to the Internet and finding the course web pages. Specific comments from the students included:

- (My use of the Internet) improved a lot. I became more comfortable with it.
- The way this class was taught was excellent -- only rarely did I get tired in class -- very interactive!
- I want to get Internet access from my home when that becomes more economical in my area. I see many uses and advantages.
- I really enjoyed learning PowerPoint and having an opportunity to get used to RDTN.

What was most interesting about the comments from the students was how each one reacted differently to each technology. Where one student really liked using the Internet, another enjoyed learning how to use the RDTN, and another liked creating presentations on PowerPoint. We recognize these students have varying learning styles and the use of various technologies provided an opportunity to use different learning activities to accommodate those learning styles.

There was also a significant delivery cost savings to NFS660. If the entire course had been delivered via RDTN the cost would have been $5,647.50. [Tab. 2]

	Hours	Cost / Hour	Total
RDTN	45	$125.50	**$5,647.50**

Table 2: Cost of NFS660 using only RDTN

The final cost of NFS660 was $1,355.41 [Tab. 3]. This does not include costs incurred by the students for the orientation session, but the students had reduced delivery fees since RDTN was not used for all the class meetings.

	Hours	Cost / Hour	Total

Orientation	19		
Exams	2		
Videos	4		
Audio-Conferencing	~11		$100.41
RDTN	10	$125.50	$1,255.00
Grand Total			**$1,355.41**

Table 3: Actual Cost of NFS660

Lessons Learned

This class has already made an impact on the way we thinking about using technology in the classroom. It affected the technology skills of the students. It appeared that they were more confident of their computer skills after this course. (One student bought a new laptop computer and actively used email to communicate with Rose following the completion of the course.)

Rose provided excellent modeling for using RDTN and PowerPoint. This was especially evident as each student presented a topic and used PowerPoint, videos or overheads to supplement their presentation. This brings up a valuable point in using technology -- modeling comfort with the technology on the instructor's part instills confidence in the student use. Not only using the technology but creating an assignment to demonstrate to these people professionally how to use the technology -- relevance is important. It was also impressive to see their final presentations in which they used the technology to provide a public workshop on maternal and infant nutrition.

Instead of tying courses to one mode of delivery, we are beginning to work from the content out and have developed a report card to help faculty understand their options. In looking at this report card, the authors discussed how the listed technologies would be used effectively at SDSU. Using a table format, we listed all the technologies currently available to faculty. Then across the top, we listed various instructional methods that our faculty use. Once we had the table constructed, we assigned letter grades as to how effective we felt these technologies would be for each method. This report card is a work in progress as improvements or enhancement are made to the technology, as the technology is made available, and as we increase instructional methods used on campus.

Where do we go from here? NFS 660 was a smashing success, both in view of the delivery method used and the final project. We were lucky in view of the time frame and student technical skills. Rose ended up creating a dynamic learning environment where technology was not a barrier.

The attitude is still prevalent that we decide on the technology first and then decide on content. NFS 660 shows us that examining the content and our student population and then adapting the technology is an effective method of delivering distance learning. At this point, very few faculty have used the variety of delivery methods as Rose did. But, many faculty are trying different delivery methods. That is an important first step. This also provides an opportunity for educational technology staff to work closely with faculty on technology and instruction techniques.

Recommendations

Putting content and student interaction first is important in classes using technology. Faculty and distance learning planners need to examine their options before committing to a particular technology. This team needs to understand their students -- where are they are on both the technology and content sides.

Choose successful faculty to pilot this concept. Our instructor was an effective teacher in the traditional classroom and flourished with her students in this class. She was able to use her interaction skills to facilitate student interaction. She also handled unexpected surprises well. Faculty at ease with the content and classroom presentation are more ready for the next step technology provides.

Have a development team available for your faculty. Although the specific support staff may move on and off the team as development progresses and needs change, a core team is an absolute necessity for faculty to make changes in their courses. Not only does the faculty member benefit by being able to concentrate on content, but they need to know someone can and will assist them. And support staff get an excellent opportunity to work closely with faculty and learn more about instructional issues from the faculty perspective.

Give yourself adequate planning time. It is important to examine objectives, content and interaction options and be willing to revamp the course to adequately support the students' learning styles.

Think efficiency. Like it or not, cost is a factor. Does it make sense to spend $360 per hour for interaction when you can do some of your activities using an alternative technology at $21 per hour?

Be sure a support team is in place during the course delivery period. Things happen. Technology does not always work. You need the staff available to fix the problem immediately. And even then, you may need an alternative plan. Keep in close contact with support people at remote sites. More than likely they have other responsibilities and are volunteering their support. They need to know how much you appreciate their help. And be sure to return the favor.

Distance learning is about learning without boundaries ... any boundaries.

Appendix A: Skills Survey

NFS 660
QUESTIONNAIRE

Please answer the following questions about your computer skills:

	Could teach someone the basics	Took a Class	Very little	No experience
Using a WWW browser (e.g. Netscape, Internet Explorer):				
Surf the Internet				
Upload/download files				
Print files				
Use the chat room				
Creating Power Point Presentations				
Viewing Power Point Presentations				
Using email				
Dialing into phone bridge (conference calls)				

During the course you will use the World Wide Web to search for information, participate in on-line discussions (chats), download files, and print files. You will also exchange email messages with others in the class. You will create a presentation (e.g., Microsoft Office Power Point) and then send that presentation by email to me.

You will need a computer(s) that has:
Microsoft Office Power Point
Connection to the Internet
WWW Browser (e.g., Netscape, Internet Explorer)
Email

Training sessions will be conducted in Brookings to be sure that you are comfortable with this new technology. We need to know what hardware and software are available to you. Please describe what is available at work and at home. (Continue on back if necessary.)

PLEASE RETURN IN ENCLOSED ENVELOPE AS SOON AS POSSIBLE.

Appendix B: Course Schedule

NFS 660
Tentative Schedule
Updated: May 14, 1997

			Type of Interaction			
Date	Hours	Topic	RDTN	Audio (Phone)	On-line Chat	Face to Face
June 13, Friday 8-5 Pugsley 214 Brookings	8	Library, Internet Resources Powerpoint, Powerpoint/Speaker Phone Presentations, RDTN, Internet Chatrooms, Evaluating journal articles (chat), Review of Pregnancy (phone)				8
June 14, Sat., 8-5 NFA 440, Brookings	8	Finish Review of Pregnancy (3 hr), Discuss abstracts on Pregnancy (1 hr), Iron Deficiency Anemia Case Study (1 hr), Review of Lactation (3 hr)				8
June 15, Dun, 8-10:30 am., Brookings	3	Discussion Lactation Abstracts (1 hr) Inborn Errors of Metabolism				3
June 23, Mon.	3	lecture by OB-GYN Doctor (1.5 hr), Presentation - (.5 hr) Infant Feeding Abstracts due	2	1		
June 30, Mon.	3	Lecture by guest (1.5 hr) Infant Feeding - (.5)	2	1		
July 7, Mon.	3	Presentation (1 hr), Review of Infant Feeding (2 hr)		3		
July 14, Mon.	3	pediatrician/HDCF lecture (1 hr), Presentation (.5 hr) Presentation (1 hr)	3			
July 21, Mon.	3	Discussion of Baby Friendly Survey (1 hr), Presentations (2 hr)		3		
July 23, Wed	3	Presentations (2 hr), Workshop Final Details (1 hr) or email		3		
July 28, Mon.	3	Workshop Presentations	3			
	2	Exams				2
Videos	4	World Food Day Lakota Breast Feeding Folic Acid - Vit of the Decade				4
Total	46 hrs		10	11	0	25

Appendix C: Delivery Evaluation

NFS 660 EVALUATION

Please rate the following types of delivery methods that have been used in the class:

	Strongly Like	Moderately Like	Neither Like nor Dislike	Moderately Dislike	Strongly Dislike
19 hours face to face instruction					
3 hours RDTN					
2 hrs RDTN, 1 hour conference call					
3 hours conference call					
email					
video on RDTN					
Slide presentation on RDTN					
Powerpoint presentation on RDTN					
video on own					

Understanding that technology and times are changing, what are the top three instructional methods that you would like to see in the next graduate class that you take? (You can use the types listed in the table or describe your own.)

1.

2.

3.

We did not get to use computer chat rooms or downloading from the website much. Would you be willing or able to use these methods in your next class?

How much did you use email?

Were you able to use the home page?

Were you able to find your grades?

Did you see the Important Messages?

Were you able to print some of the web pages?

Did this class improve your use of the internet a little or a lot or not at all?

Comments:

Please return to NFS Dept, PO Box 2275A, SDSU, Brookings, SD 57007-0497 asap.

The Interactive Mathematics Laboratory

William M. Farmer
Microcomputer Studies Program
Department of Statistics
St. Cloud State University
720 Fourth Avenue South
St. Cloud, MN 56301-4498, USA
wmfarmer@stcloudstate.edu

Abstract

An *interactive mathematics laboratory* (IML) is a computer system with a set of integrated tools designed to facilitate a wide range of mathematical activity. It is an *interactive* environment where the user can create and explore rigorous mathematics. It is also a *mechanized* environment where the system can perform many functions useful to the user. The IML has the potential to transform how people learn and practice mathematics. IMLs do not exist today, but much of the technology needed for building an IML can be found in contemporary theorem provers and computer algebra systems.

The Interactive Mathematics Laboratory

William M. Farmer
Microcomputer Studies Program
Department of Statistics
St. Cloud State University
720 Fourth Avenue South
St. Cloud, MN 56301-4498, USA
wmfarmer@stcloudstate.edu

1 Introduction

Computer technology has transformed the practice of many disciplines. Even though much of mathematics is strongly formal and mechanical, mathematics has largely been passed over by the information age. True, computers and calculators have revolutionized how mathematical computations are performed, but the central process of creating mathematical models and exploring them by stating and proving conjectures is still done with little more than pencil and paper.

As mathematics enters the 21st Century, this picture will drastically change with the arrival of a new instrument for mathematics: the *interactive mathematics laboratory* (IML). An IML is a kind of *mechanized mathematics system* (MMS). MMSs include computer algebra systems, theorem provers, and hardware and software verification systems. In contract to other MMSs, an IML is designed to facilitate a wide range of mathematical activity. It is an *interactive* environment where the user can create and explore rigorous mathematics. It is also a *mechanized* environment where the system can perform many functions useful to the user. IMLs do not exist today, but less powerful MMSs have been developed and tested. If there is sufficient support for the IML, it could make its appearance sometime in the next decade.

The IML will transform how people do mathematics. It will extend the mathematical reach of engineers and scientists, allowing them to use more mathematics in their work and to use it better. It will enable high school and college students to participate in the creative/explorative process of mathematics which is rarely experienced firsthand by students today. And the IML may even change the way mathematicians conduct research.

The purpose of this paper is to describe what an IML is and how it will change mathematics practice. It is intended to advance the cause of mechanized mathematics and to help prepare educators, mathematicians, software developers, scientists, engineers, and research sponsors for a new age of mathematics.

2 Mathematics Practice Today

When people of the 21st Century look back at mathematics practice of the 20th Century, two facts will stand out. First, the tools for mathematical reasoning commonly used in the 20th Century were exceedingly primitive. Second, the mathematical knowledge base in the 20th Century was not readily accessible to ordinary mathematics practitioners.

2.1 Primitive Tools

Today mathematics practice is dominated by three major activities:

(1) Formulating mathematical ideas.

(2) Performing mathematical calculations.

(3) Reading and writing mathematical texts.

The first activity includes the formulation of models, definitions, conjectures, and proofs. Mathematical calculations are performed using calculation devices such as hand-held calculators, general-purpose programming languages, and computer algebra systems. Mathematical texts describe mathematical ideas expressed informally in natural language.

Mathematics practitioners formulate mathematical ideas using just their minds and a few primitive tools like pencil and paper. Although calculation devices are often used for exploring mathematical ideas, the process of organizing mathematical concepts and making logical inferences is routinely performed without the assistance of sophisticated information processing tools. Theorem provers – computer systems for performing deductions and developing proofs – are available, but few people actually use them for doing real mathematics. Since they usually require the user to work at an uncomfortably low level of abstraction, they are difficult to learn and burdensome to use.

Computer-based calculation devices have completely revolutionized how people calculate – the abacus and the slide rule are now entirely obsolete. Although they perform calculations at terrific speeds, they are often based on primitive and inflexible principles which may actually violate the rules of rigorous mathematics. For example, computer algebra systems are not designed for calculations that depend on context [Farmer, Guttman, & Thayer 1995], and so, for instance, they return the wrong answer for the integration $\int x^n dx$ when $n = -1$. Computer algebra systems are in general untrustworthy, and the results they return must be carefully checked before they can be used as part of a mathematical proof. Another sobering example is the popular programming language Visual Basic which will return an overflow error when evaluating the simple expression $1000 * 1000$.

Mathematical texts are read and written using ordinary or mathematical text editing systems. The mathematical text editing systems like TeX are only slightly more sophisticated than ordinary systems. They are designed to manipulate words and paragraphs instead of mathematical objects like definitions and theorems. Special languages such as first-order predicate logic for expressing and manipulating mathematical ideas as formal objects have been devised, but they are often based on principles that clash with mathematical practice and are impractical without good mechanical support.

2.2 Inaccessible Knowledge Base

Mathematical models, concepts, and facts are interconnected to a degree that astonishes even seasoned mathematicians. The rich interconnectedness of mathematics is a principal hallmark of mathematics. The created ideas and discovered facts of mathematics form a great, highly structured edifice of knowledge. Infinitely fascinating and beautiful, this *Magnificent Matrix* is the product of many centuries of mathematical activity. It is the playground and the workplace of mathematics – a constantly growing world that is too big and too intricate to be understood by one person or even one culture.

Where does the Magnificent Matrix reside? Bits and pieces of it are in mathematics textbooks and journal articles. Some of it is embodied in such things as engineers' designs, programmers' code, and scientists' experiments. Some also exists in unpublished notes that are not available to the public. However, the majority of the Magnificent Matrix resides nowhere else than in the minds of mathematicians.

The last point may seem a bit fantastic to some readers. It deserves an example. Consider a journal paper written by a mathematician about some area of mathematics. Let us suppose that the paper contains the statements of several theorems with sketches of their proofs. The sketch for one proof is simply the phrase "By induction". Such a brief proof sketch suggests that the author of the paper expects that mathematicians knowledgeable about the paper's subject matter will be able to easily see how to prove the theorem using the principle of induction. Some conscientious readers might actually write down a proof, but most knowledgeable readers would not feel the need to produce a full proof. It is very possible that a proof has never been written down, even by the author herself. This may not be a problem for the group of people who are knowledgeable about the paper's subject matter, a group which might well be just a handful of people across the whole world. But for the rest of humanity (and even for most mathematicians) this proof could be as inaccessible as the daydreams people had in prehistoric times.

The crucial details of the Magnificent Matrix – how ideas are connected and why inferences are true – are largely missing. These details are essential for a genuine understanding of mathematics. With the right training, intelligence, and persistence these details can be rediscovered. Mathematicians are very good at rediscovering mathematical details, but most other people are not. As a consequence, mathematics is a difficult subject for most people. For them learning mathematics is like learning how a computer program works when one has access to a vague description of it but not to its actual code.

Mathematics could be a much easier subject to learn and to use if there were a library containing the creations and discoveries that form the Magnificent Matrix. No such library exists. Even all the mathematical information on the World Wide Web taken together does not come very close to being a library of this kind.

One reason for this is that most mathematical knowledge that is available in books, journals, and on the Web is *static*. Static mathematics is mathematics described as a collection of fixed individual concepts and facts. It is like a path through a garden where one is not allowed to leave the path or to explore other paths. Static mathematics is easy to present with tradition media, but it is a false representation of the *dynamic* nature of mathematics.

Dynamic mathematics is mathematics described so that people can creatively explore it. It is like a garden where a visitor can choose her own path. Consider a theorem with a proof in an ordinary mathematics book. The book offers the reader the opportunity to study the theorem and the proof as they are presented, but the book is not designed to help the reader to explore variations of the theorem and the proof. If the theorem and the proof were presented in a dynamic medium, the reader could modify the theorem and then try "running" the proof to see if the new version of the theorem were true. And, if the old proof failed on the new version, the reader could modify it and run the modified proof on the new version.

Researchers in the field of mechanized mathematics have developed experimental dynamic libraries of mathematics. The most extensive of these is the mathematical library created by the Mizar project [Rudnicki 1992]. Unfortunately, these libraries are relatively small and highly inaccessible to ordinary mathematics practitioners.

A great deal of dynamic mathematics is contained in calculation devices such as hand-held calculators, programming languages, and computer algebra systems. These devices are very powerful tools for creating and exploring *algorithmic mathematics*, the mathematics of calculation. However, they are not designed for creating and exploring *axiomatic mathematics*, the mathematics of deduction. Moreover, they operate as "black boxes" in which the calculation process is completely hidden from view. They return the result of a calculation, but they neither describe how the calculation was performed nor show why the calculation is correct.

In recent years a proposal called " The QED Manifesto" [anonymous], [anonymous 1994] has been put before the public. The proposal calls for the construction of a "computer system that effectively represents all important mathematical knowledge and techniques", that "conform[s] to the highest standard of mathematical rigor", and that "include[es] the use of strict formality in the internal representation of knowledge and the use of mechanical methods to check proofs of the correctness of all entries in the system". Needless to say, the QED project would be a monumental undertaking that would require the cooperation of a great many people in science and government.

Although a few hundred researchers, mostly in the field of automated reasoning, have expressed support for the project, QED has failed to capture the public's interest. This is due perhaps in part to the nature of the proposal itself. With its strong emphasis on formality and correctness, QED calls for something that is more like an archive for a select group of scholars than a library for the general public. QED is too much like a monument for the glorification of mathematics and not enough like a public garden where people can go to explore and create mathematical ideas.

As long as the secrets of the Magnificent Matrix are largely inaccessible to the public, mathematics will remain an underappreciated and poorly understood discipline.

3 IML Components and Services

Most people today write papers and books using a text editing system that runs on a computer. These systems offer an environment which facilitates many activities involved with writing such as entering text, checking spelling, formatting, cutting and pasting, viewing text, and searching for character strings. A text editing system makes writing easier, but it does not substitute for a writer's creativity. Nevertheless, text editing systems have transformed how people write.

The purpose of an IML is to support the main mathematical activities including formulating models, making definitions, proving conjectures, performing calculations, and searching for related information. IMLs assist one in doing mathematics in a similar way to how text editing systems assist one in writing. They also allow one to practice mathematics in a similar way to how a language laboratory allows one to practice a foreign language. Like a text editing system, an IML is an instrument to enhance one's creativity, not replace it.

Although there is certainly more than one possible design for an IML, most designs would consist of three principal components:

(1) A *theory library*, composed of interlinked axiomatic theories, which is intended to serve as a database of mathematics.

(2) A *reasoning engine* composed of three highly integrated facilities:

  (a) A *theory developer* for extending the theory library.

  (b) A *prover* for testing whether conjectures are true.

  (c) A *calculator* for performing calculations.

(3) A *user interface* that will enable users to interact with the IML in different styles.

An IML with these components would provide a variety of services for both students just beginning to learn mathematics as well as highly trained engineers, scientists, and mathematicians. Some examples of these services follow.

Theory library The theory library of an IML would be a formal, digital description of the Magnificent Matrix. Each theory in the library would offer a different partial view of the Magnificent Matrix. The information stored in the library would be highly dynamic in the sense given above and would include both algorithmic and axiomatic mathematics. The theories would be linked via *theory interpretations* [Enderton 1972], [Farmer 1994], [Schoenfield 1967], which would serve as conduits through which information from one theory could be "transported" to another theory [Barwise & Seligman 1997]. The theory library would be a distributed library with part residing on the World Wide Web and part residing locally on each individual IML.

Exploration An IML would provide the ability to "browse" the theory library in sophisticated ways. For example, the user could ask to see all the different ways of defining a continuous function and then explore what impact these different definitions have on the basic development of calculus. A user could ask what a certain complex expression reduces to in some particular algebraic theory. After choosing a well-known theorem, a user could ask for other theorems in other theories that are generalizations of the it. And a user could study a selected proof to whatever level of detail is desired.

Creation An IML would facilitate the creation of mathematical ideas and objects, including mathematical models, definitions, conjectures, proofs, programs, notation, axiomatic theories, and theory interpretations. The user could start from scratch or build on the work of others. The creations could be added to the theory library, and if desired, made available to the public. This facility could be used by mathematicians to produce genuinely new mathematics and by students to learn how mathematics is created.

Mathematical proof An IML would support the development of mathematical proofs. Proof is what separates mathematics from all other disciplines; it is the prime instrument of the creative/explorative process of mathematics. IML-produced proofs would be both humanly comprehensible like the informal proofs of mathematicians and mechanically manipulatable like the formal proofs of logicians. While many educators believe that proofs should be de-emphasized today in mathematics education, an IML would offer a new kind of proof that is easier to understand and to construct.

Calculation Like a computer algebra system, an IML would be able to perform calculations using both symbolic and numeric computation. However, unlike a contemporary computer algebra system, an IML would perform its calculations within a rigorous logical framework. Moreover, many calculations could be performed in a "transparent" mode in which the steps of the calculation would be visible to the user.

Organizational support An IML would automatically handle the lion's share of the organizational concerns and the drudgework involved in mathematical reasoning. This, in turn, would allow the user to concentrate more fully on the key concepts and techniques being studied or applied. It would also enable students to work on more realistic and better focused problems.

Logical support The rules of rigorous reasoning would be effectively encoded in an IML, and the system would check the soundness of the operations performed by the user. Consequently, the user would have direct access to the "rules of the game", and all attempts to make unfounded conclusions would be immediately identified.

Context management An IML would keep track of the context the user is working in, i.e., the assumptions and definitions that are currently in force [Farmer, Guttman, & Thayer 1995]. The user would have the freedom to examine the contents of the context and to change the context by switching to a new set of assumptions and definitions. One of the things that makes traditional mathematics difficult is that when one is using or studying mathematics the context at issue is usually not explicit and often only a trained mathematician will have a detailed understanding of it.

Automated reasoning Automated reasoning would be utilized systematically in an IML, particularly to handle low-level details. This would allow the user to focus her reasoning on the key aspects of a problem and let the system handle the more routine aspects. The level of automated reasoning could be controlled, which would be particularly important for students. For each student, an IML would establish a demarcation between what is routine and what is key. The system would largely handle what is routine, while the student, with only limited help from the system, would be responsible for what is key. The frontier between what is routine and what is key – and what is automated and what is not – would be moved forward as the student mastered new concepts and techniques.

Notational freedom Since an IML manipulates and stores mathematics presented in a precise, formal matter, it would be easy to display the mathematics using multiple notations. The user could choose whatever notation she preferred, and mathematics developed by one user in one notation could be viewed by another user in another notation.

4 Related Technologies

We have mentioned above that IMLs do not currently exist. However, much of the technology needed for building an IML can be found in contemporary MMSs. The major computer algebra systems, such as Macsyma[1] [Macsyma Inc. 1996], Maple[2] [Char, Geddes, Gonnet, Leong, Monagan, & Watt 1991], and Mathematica[3] [Wolfram 1991], contain a king's treasure of techniques for performing mathematical calculations. Theorem provers for interactively discovering and developing proofs are a more important source of technology. Some of the major examples are Coq [Barros, et al. 1997], EVES [Craigen, et al. 1991], HOL [Gordon & Melham 1993], IMPS [Farmer, Guttman, & Thayer Fábrega 1996], [Farmer, Guttman, & Thayer 1993], Isabelle [Nipkow & Paulson 1992], Mizar [Rudnicki 1992], NQTHM [Boyer & Moore 1988], Nuprl [Constable, et. al 1986], Otter [McCune 1990], and PVS [Owre, Rusby, & Shankar 1992]. These systems have much of the technology that an IML needs, but they are more narrow in scope than an IML and are very difficult to use without a fairly deep understanding of mechanized mathematics.

The IMPS system is a good example to look at. It has the basic components of an IML and is intended to provide mechanical support for traditional mathematical techniques and styles of practice. Moreover, IMPS is the main source of inspiration for our notion of an IML.

IMPS has a theory library containing a relatively large amount and variety of basic mathematics. It includes formalizations of the real number system and objects like sets and sequences; theories of abstract mathematical structures such as groups and metric spaces; and theories to support specific applications of IMPS in computer science. Unlike the theory libraries of most other theorem provers, the theories in the IMPS library are linked by theory interpretations which provide the infrastructure for the "little theories" style of the axiomatic method [Farmer, Guttman, & Thayer 1992]. The little theories approach is a powerful technique for capturing the great interconnectedness of mathematics.

IMPS users can develop proofs interactively with the system. The proofs are a blend of calculation and deduction. Low-level inferences can be performed with calculation tools that simplify expressions and apply theorems. High-level inferences are performed by applying proof commands or by executing "proof scripts" that apply a sequence of proof commands in an intelligent manner [Farmer, Guttman, Nadel, & Thayer 1994]. IMPS makes a significant step toward uniting algorithmic and axiomatic mathematics.

The IMPS logic is a version of simple type theory [Church 1940], called LUTINS[4] [Farmer 1990], [Farmer 1993], [Farmer 1994]. Equipped with special machinery for reasoning about functions, it has proven to be highly effective for formalizing classical mathematics. LUTINS is exceptional in that partial functions like division and undefined terms like 2/0 can be represented and manipulated without compromising the basic principles of classical predicate logic.

IMPS does not have the full capabilities of an IML: it supports only a modest set of mathematical activities and requires a high degree of mathematical sophistication to use effectively. For example, IMPS has a powerful mechanism for defining functions by recursion which is based on the use of monotone functionals. The user must have a technical understanding of this mechanism to use it for even simple recursive definitions. However, it is quite conceivable that a suitable layer of user interface software could be developed which would insulate the user from the technicalities of the mechanism. Contemporary theorem provers like IMPS employ very little user interface software of this kind. To be accessible to ordinary people, an IML must contain a user interface that is much more powerful and sophisticated than the theorem prover user interfaces that we see today.

IMPS is a significant step towards an IML. It demonstrates the feasibility of the IML and the impact the IML will have on mathematics practice. It also demonstrates that the usefulness of an IML will depend at

[1] Macsyma is a registered trademark of Macsyma Inc.
[2] Maple is a registered trademark of Waterloo Maple Software.
[3] Mathematica is a registered trademark of Wolfram Research, Inc.
[4] Pronounced as the word in French.

least as much on good system engineering and user interface software as on good logical and deductive machinery.

5 Obstacles

Several obstacles stand in the way of building an IML that can become an integral part of mathematics practice. Some of the most important of them follow.

High cost The cost of developing a state-of-the-art IML that is accessible to ordinary students and professionals would be very high, on the order of millions of U.S. dollars. A development team for an IML would require funding for an extended period of time and would need expertise in mathematics, logic, automated reasoning, mathematical computation, human-computer interaction, and education.

Lack of interest in the mathematics community The mathematics community has largely ignored the field of mechanized mathematics. Only a miniscule fraction of mathematicians are actively involved in developing or applying MMSs. Considering the potential impact that the IML could have on how people learn and practice mathematics, it is most surprising that mathematicians are not in the vanguard of mechanized mathematics. Their knowledge and leadership is essential for the development and deployment of an IML.

Inaccessibility Contemporary MMSs are essentially only accessible to highly sophisticated users – typically people who have graduate degrees in computer science or mathematics. Developing an IML that is usable by and useful to ordinary mathematics practitioners is an important area for research.

System engineering To be effective, the various subsystems of an IML (e.g., user interface, logic, prover, and theory development facility) must be highly integrated like the component systems of an automobile. System developers in the field of mechanized mathematics have relatively limited experience in the kind of system engineering that is needed for an IML. This is another important area for research.

Underlying logic The great majority of contemporary MMSs are based either on simple logics which are too inexpressive or on type theories which are too esoteric for most typical mathematics practitioners. To be successful, an IML needs an underlying logic which is highly expressive, familiar to users, and well understood by developers. The author has argued in [Farmer 1998] and [Farmer & Guttman 1998] in favor of a particular version of set theory that has the same convenient machinery for reasoning about functions as LUTINS, the logic of IMPS.

6 Conclusions

A few hundred years ago mathematicians began using symbolic expressions to describe mathematical entities and to assert mathematical relationships. This use of symbols has been a powerful transformational force in mathematics practice. It has made mathematics easier to do and has allowed people to obtain a stronger and deeper grasp of mathematics.

The IML will have a similar effect on mathematics practice. It will unlock the Magnificent Matrix and enable more people to become mathematically engaged. It will bring mathematics, finally, into the information age.

The promise of the IML for mathematics is certainly great. But to achieve this promise will require a concerted effort by many people of multiple disciplines. Research sponsors have to provide better funding for mechanized mathematics. Mathematicians have to get involved in the development and application of MMSs. Software developers have to produce new technology for making MMSs more effective and easier to use. Scientists and engineers have to integrate MMSs into their work. And educators have to restructure the mathematics curriculum – for example, by placing more emphasis on axiomatic mathematics – to prepare students for the arrival of the IML and a new age of mathematics.

References

[1] [anonymous] anonymous. The QED project. Available at `http://www.mcs.anl.gov:80/qed/`.

[2] [anonymous 1994] anonymous (1994). The QED manifesto. In A. Bundy, editor. *Automated Deduction – CADE-12*, volume 814 of *Lecture Notes in Computer Science*, 238-251. Springer-Verlag.

[3] [Barros, et al. 1997] Barros, B., et al (1997). The Coq proof assistant reference manual, version 6.1. Available at `ftp://ftp.inria.fr/INRIA/coq/V6.1/doc/Reference-Manual.dvi.gz`.

[4] [Barwise & Seligman 1997] Barwise, J.; & Seligman, J (1997). *Information Flow: The Logic of Distributed Systems*, volume 44 of *Tracts in Computer Science*. Cambridge University Press.

[5] [Boyer & Moore 1988] Boyer, R.; & Moore, J. (1988). *A Computational Logic Handbook*. Academic Press.

[6] [Char, Geddes, Gonnet, Leong, Monagan, & Watt 1991] Char, B. W.; Geddes, K. O.; Gonnet, G. H.; Leong, B. L.; Monagan, M. B.; & Watt, S. M. (1991). *Maple V Language Reference Manual*. Springer-Verlag.

[7] [Church 1940] Church, A. (1940). A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56-68.

[8] [Constable, Allen, Bromley, Cleaveland, Cremer, Harper, Howe, Knoblock, Mendler, Panagaden, Sasaki, & Smith 1986] Constable, R. L.; Allen, S. F.; Bromley, H. M.; Cleaveland, W. R.; Cremer, J. F.; Harper, R. W.; Howe, D. J.; Knoblock, T. B.; Mendler, N. P.; Panagaden, P.; Sasaki, J. T.; & Smith, S. F. *Implementing Mathematics with the Nuprl Proof Development System.* Englewood Cliffs, NJ: Prentice-Hall.

[9] [Craigen, Kromodimoeljo, Meisels, Pase, & Saaltink 1991] Craigen, D.; Kromodimoeljo, S.; Meisels, I.; Pase, B.; & Saaltink, M. (1991). EVES: An overview. Technical Report CP-91-5402-43, ORA Corporation.

[10] [Enderton 1972] Enderton, H. B. (1972). *A Mathematical Introduction to Logic*. Academic Press.

[11] [Farmer 1990] Farmer, W. M. (1990). A partial functions version of Church's simple theory of types. *Journal of Symbolic Logic*, 55:1269-91.

[12] [Farmer 1993] Farmer, W. M. (1993). A simple type theory with partial functions and subtypes. *Annals of Pure and Applied Logic*, 64:211-240.

[13] [Farmer 1994] Farmer, W. M. (1994). Theory interpretation in simple type theory. In J. Heering et al., editor, *Higher-Order Algebra, Logic, and Term Rewriting*, volume 816 of *Lecture Notes in Computer Science*, 96-123. Springer-Verlag.

[14] [Farmer 1998] Farmer, W. M. (1998). STMM: A Set Theory for Mechanized Mathematics. Submitted for publication.

[15] [Farmer & Guttman 1998] Farmer, W. M.; & Guttman, J. D. (1998). A set theory with support for partial functions. Presented at the Workshop on Dynamic Logic, Epistemic Logic, and Partial Logic, Montreal, Canada, June 9-12, 1995. Submitted for publication. Available at `ftp://math.harvard.edu/imps/doc/`.

[16] [Farmer, Guttman, & Thayer Fábrega 1996] Farmer, W. M.; Guttman, J. D.; & Thayer Fábrega, F. J. (1996). IMPS: An updated system description. In M. McRobbie and J. Slaney, editors, *Automated Deduction – CADE-13,* volume 1104 of *Lecture Notes in Computer Science,* 298-302. Springer-Verlag.

[17] [Farmer, Guttman, Nadel, & Thayer 1994] Farmer, W. M.; Guttman, J. D.; Nadel, M. E.; & Thayer, F. J. (1994). Proof script pragmatics in IMPS. In A. Bundy, editor, *Automated Deduction – CADE-12*, volume 814 of *Lecture Notes in Computer Science,* 356-370. Springer-Verlag.

[18] [Farmer, Guttman, & Thayer 1992] Farmer, W. M.; Guttman, J. D.; & Thayer, F. J. (1992). Little theories. In D. Kapur, editor, *Automated Deduction – CADE-11,* volume 607 of *Lecture Notes in Computer Science,* 567-581. Springer-Verlag.

[19] [Farmer, Guttman, & Thayer 1993] Farmer, W. M.; Guttman, J. D.; & Thayer, F. J. (1993). IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning,* 11:213-248.

[20] [Farmer, Guttman, & Thayer 1995] Farmer, W. M.; Guttman, J. D.; & Thayer, F. J. (1995). Contexts in mathematical reasoning and computation. *Journal of Symbolic Computation*, 19:201-216.

[21] [Gordon & Melham 1993] Gordon, M. J. C.; & Melham, T. F. (1993). *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic.* Cambridge University Press.

[22] [Macsyma Inc. 1996] Macsyma Inc. (1996). *Macsyma Mathematics and System Reference Manual.* Macsyma Inc.

[23] [McCune 1990] McCune, W. (1990). OTTER 2.0. In M. E. Stickel, editor, *10th International Conference on Automated Deduction,* volume 449 of *Lecture Notes in Computer Science,* 673-676. Springer-Verlag.

[24] [Nipkow & Paulson 1992] Nipkow, T.; & Paulson, L. C. (1992). Isabelle-91. In D. Kapur, editor, *Automated Deduction – CADE-11,* volume 607 of *Lecture Notes in Computer Science,* 673-676. Springer-Verlag.

[25] [Owre, Rushby, & Shankar 1992] Owre, S.; Rusby, J. M.; & Shankar, N. (1992). PVS: A prototype verification system. In D. Kapur, editor, *Automated Deduction – CADE-11,* volume 607 of *Lecture Notes in Computer Science,* 748-752. Springer-Verlag.

[26] [Rudnicki 1992] Rudnicki, P. (1992). An overview of the MIZAR project. Technical Report, Department of Computing Science, University of Alberta.

[27] [Schoenfield 1967] Schoenfield, J. R. (1967). *Mathematical Logic.* Addison-Wesley.

[28] [Wolfram 1991] Wolfram, S. (1991). *A System for Doing Mathematics by Computer.* Addison-Wesley.

Acknowledgements

Many of the ideas in this paper grew out of conversations I have had with Dr. Joshua Guttman, Dr. Javier Thayer Fábrega, and Dr. Nadel in the course of developing IMPS.

Student Technology Teams: A Panel

Rhonda Ficek
CSIS Department
Moorhead State University
Moorhead, MN 56560
ficek@mhd1.moorhead.msus.edu

Dr. James Ross
Information Technology Services
North Dakota State University
Fargo, ND 58105
ross@plains.nodak.edu

Elizabeth Smith
Information Technology Services
North Dakota State University
Fargo, ND 58105
elsmith@badlands.nodak.edu

A panel discussion on student technology teams will focus on the following issues related to utilizing students to assist with technology support.

- Recruitment of students
- Training of the student technology teams
- Obtaining funding for student technology teams
- Tasks assigned to student technologists:
    - Assisting with hands-on training sessions conducted by faculty FOR faculty/staff
    - Leading hands-on training sessions for students
    - Faculty/student mentorships (to assist faculty in creating online materials, using web-based research projects, etc.)
    - Creating/maintaining web sites for the campus

Measuring Learning Outcomes in a Competency-Based CS 2 Course

Sue Fitzgerald
Department of Information and Computer Sciences
Metropolitan State University
Minneapolis-St. Paul, MN USA
FitzgeraldS@msus1.msus.edu

Abstract

This paper presents a unique approach to learning assessment and reporting. A connection has been established between a) those qualities identified by outside accrediting and legislative bodies and our university as being universally desirable and b) the individual learning outcomes for discipline-specific courses. Data collected using this approach makes it possible to see how well the computer information systems and computer science majors offered at Metropolitan State University provide coverage of the quality factors of higher order thinking, scientific and quantitative literacy, readiness for work and career, and communication skills, according to the assessment of the those actually doing the teaching. It is also possible to see how well students in key classes are meeting the outcomes associated with specific quality factors over time. This in turn permits the department to adjust and strengthen the program. Data collected during the spring and fall quarters of 1997 is presented here.

Measuring Learning Outcomes in a Competency-Based CS 2 Course

Sue Fitzgerald
Department of Information and Computer Sciences
Metropolitan State University
Minneapolis-St. Paul, MN USA
FitzgeraldS@msus1.msus.edu

Abstract

This paper presents a unique approach to learning assessment and reporting. A connection has been established between a) those qualities identified by outside accrediting and legislative bodies and our university as being universally desirable and b) the individual learning outcomes for discipline-specific courses. Data collected using this approach makes it possible to see how well the computer information systems and computer science majors offered at Metropolitan State University provide coverage of the quality factors of higher order thinking, scientific and quantitative literacy, readiness for work and career, and communication skills, according to the assessment of the those actually doing the teaching. It is also possible to see how well students in key classes are meeting the outcomes associated with specific quality factors over time. This in turn permits the department to adjust and strengthen the program. Data collected during the Spring and Fall quarters of 1997 is presented here.

Key words: assessment, computer science education, competency-based education, CS 2, learning outcomes

Background

In 1989, the Commission on Institutions of Higher Education began an initiative to document student academic achievement. It called upon colleges and universities to structure assessment plans consistent with their stated missions and goals. The purpose of such assessment plans is to strengthen teaching and learning. Not only must student achievement be assessed, but the assessment plan and its implementation must also be tied to the institution's long-range strategic planning, providing information essential to improving the institution as a whole. Specifically, the institution is required to link the assessment plan to its mission; it must provide evidence that the faculty has participated in the development of the assessment plan; the plan must lead to institutional improvement; the timeline for the plan must be appropriate and realistic; and the plan must be appropriately administered.

This emphasis on assessment from our accrediting bodies is coupled with a current political climate in which state legislatures are focusing on outcomes assessment and legislating measurable results from higher education. In the current rapidly evolving job market which increasingly requires highly skilled, technologically savvy workers, industry representatives complain that workers lack specific job skills, often targeting emerging technologies or software packages that have a shelf-life of less than three years. As state legislators attempt to respond to these complaints and address questions about the effective use of public moneys for educational purposes, they increasingly hold institutions of higher education accountable for students' preparedness to enter the job market.

As a result of these influences, most institutions are struggling with developing assessment plans that assess learning and provide measurable results. Many have no experience with formal assessment techniques and few institutional structures for evaluating such data as are available.

This paper addresses the institutional approach taken by Metropolitan State University and uses data taken from the 1997 Spring and Fall quarters of ICS 360 Computer Algorithms I (CS 2 equivalent) for illustration.

Competency-based Assessment

Historically, Metropolitan State University's educational philosophy has been competency-based, with a focus on identifying desired student outcomes and assessing the student's competence based on these criteria. This approach is particularly effective for adult learners who are most interested in increased professional competence. Each student is given a learning evaluation form, which includes a competency statement and specific learning outcomes for each course. The manner in which each learning outcome is to be evaluated is indicated (e.g., tests, programs, papers, etc.). In addition to a final course grade, each student is evaluated based on each learning outcome. An example of a learning evaluation form containing a competency statement and outcomes for a CS 2 class is shown below.

Sample Competence Statement and Learning Outcomes for ICS 360 Computer Algorithms I: A CS 2 Course

Competence statement:
Knows concepts and approaches to the implementation of lists, stacks and queues well enough to write moderately difficult programs using iteration, recursion, arrays, records and pointers.

Evaluation:
Design and implementation (Q5, Q6, Q9): Is able to produce designs and translate them into robust, error-free, readable, well-documented code that follows accepted programming style guidelines including modularity. Demonstrated through programming assignments.

Problem solving skills (Q2): Is able to solve moderately difficult logical problems. Can evaluate and modify existing algorithms as demonstrated through problem solving exercises and tests.

Communication (Q8): Is able to document designs and explain common computer science concepts as demonstrated through program design documentation and exams.

Reliability (Q1): Attends class, turns in work and meets deadlines.

Outcomes Assessment Using Quality Factors

With its history of competency-based education and a tradition of evaluating specific outcomes for each course, it seemed natural for Metro State to build its assessment program around the learning evaluation form.

An additional component was also needed to organize individual course outcomes in such a fashion that trends or composite data could easily be discerned. Specific learning outcomes for graduates of all programs at Metro State were identified, based on a 1993 Minnesota State Blue Ribbon Commission. These quality factors reflect the specific concerns of legislators and employers. Metro State's quality factors, taken from the 1996-1998 Metropolitan State University, p. 4-5, are listed below:

Q1. Preparedness. Each entering student will have met the university's admission standards and will have completed a diagnostic assessment procedure, identified educational strengths and deficiencies, and addressed any deficiencies.

Q2. Higher Order Thinking. Students will demonstrate higher order thinking in a variety of ways throughout the educational process as well as by completing a final integrating experience.

Q3. Global Understanding. Students will demonstrate progress toward acquisition of a global perspective through international study or the study of foreign language and culture, and will demonstrate the ability to articulate the interrelationships of world economics, environment, geography, history, politics, religion and the arts.

Q4. Multicultural Perspective. Students will demonstrate the knowledge, skills and values gained by the study of those groups in the United States oppressed by virtue of race, gender and class by including learning related to multiculturalism as part of their baccalaureate learning experiences and by working together in diverse learning communities toward shared goals.

Q5. Scientific and Quantitative Literacy. Students will demonstrate quantitative and scientific literacy, including an understanding of the transforming role of technology in world society.

Q6. Readiness for Work and Career. Students will demonstrate readiness for a chosen vocation by integrating theoretical and practical learning in an appropriate setting, depending upon educational goals and previous experiences.

Q7. Responsible Citizenship in a Democracy. Students will demonstrate an understanding of citizen interdependence and the importance of citizen involvement in a democracy through successful completion of a community service, citizen participation, or social action project. Additionally, students should develop the ability to articulate the standards of ethical behavior expected of them and of others in their professional, public and personal lives.

Q8. Communication. Students will demonstrate the ability to effectively communicate through a variety of written, oral and nonverbal means with individuals and in groups.

Q9. Discipline Knowledge. Students should demonstrate in-depth knowledge of a specific discipline or subject area. This might include interdisciplinary knowledge.

During the 1996-1997 academic year, a massive project was undertaken university-wide to “Q” the learning evaluation forms. Faculty members were encouraged to examine the ways in which their courses meet broader university outcomes. Each faculty member was asked to specify the quality factors associated with the specific course outcomes stated on the learning evaluation forms for their classes.

Based on the learning outcomes previously identified for ICS 360 Computer Algorithms I, the quality factors of preparedness (Q1), higher order thinking (Q2), scientific and quantitative literacy (Q4), readiness for work and career (Q6), communication (Q8), and discipline knowledge (Q9), were identified as goals for our CS 2 course.

Results

Data was collected for the CS 2 course outcomes during the 1997 Spring and Fall quarters. The first piece of information provided by this approach to assessment is the clear identification of the quality factors targeted by each class. As mentioned above, the quality factors of preparedness (Q1), higher order thinking (Q2), scientific and quantitative literacy (Q4), readiness for work and career (Q6), communication (Q8), and discipline knowledge (Q9), were identified as goals for our CS 2 course.

The quality factor of preparedness for college (Q1) was measured based on the student's pattern of reliably turning in work, meeting deadlines and attending class. Higher order thinking skills (Q2) were assessed based on the student's ability to solve homework and test problems involving logic and problem solving. The quality factors of scientific and quantitative literacy (Q5), readiness for work and career (Q6), and discipline knowledge (Q9) were assessed through programming assignments, including evaluation of programming style, user interface design, and code robustness. Design documentation and essay test questions determined assessment of communication skills (Q8).

[Fig. 1-6] show the distribution of students ranked Excellent, Good, Adequate, Partially Adequate, and Inadequate for each quality factor measured. Spring and Fall 1997 outcomes are compared. The corresponding raw data for numbers of students ranked at each level is presented in [Tab. 1-2]. Assessments were performed only on those students who completed the class.

The distribution of the assessment rankings (Excellent, Good, Adequate, Partially Adequate, Inadequate) for the selected quality factors clearly identifies any targeted areas which have not been adequately addressed by the class. Since 80% or more of students were ranked at least at the Adequate level for all selected quality factors, no major problems seem to be indicated here.

[Tab. 3] summarizes the percentages of students who were assessed at the level of adequate or better for each quality factor. This cumulative data represents student success levels simply and clearly, partially satisfying the requirements of our accrediting and legislative bodies for measurable outcomes.

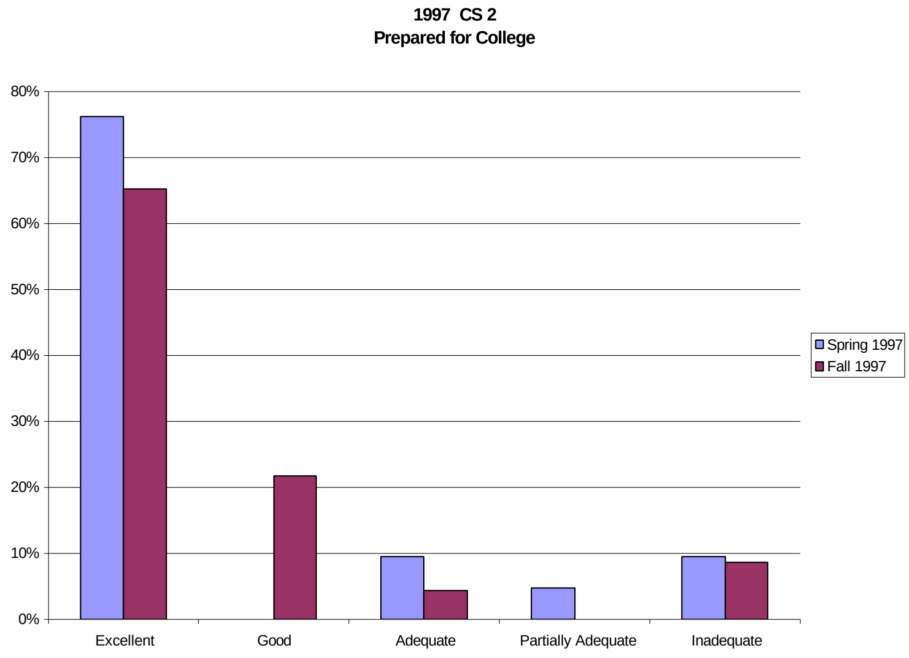


Figure 1: Prepared for College

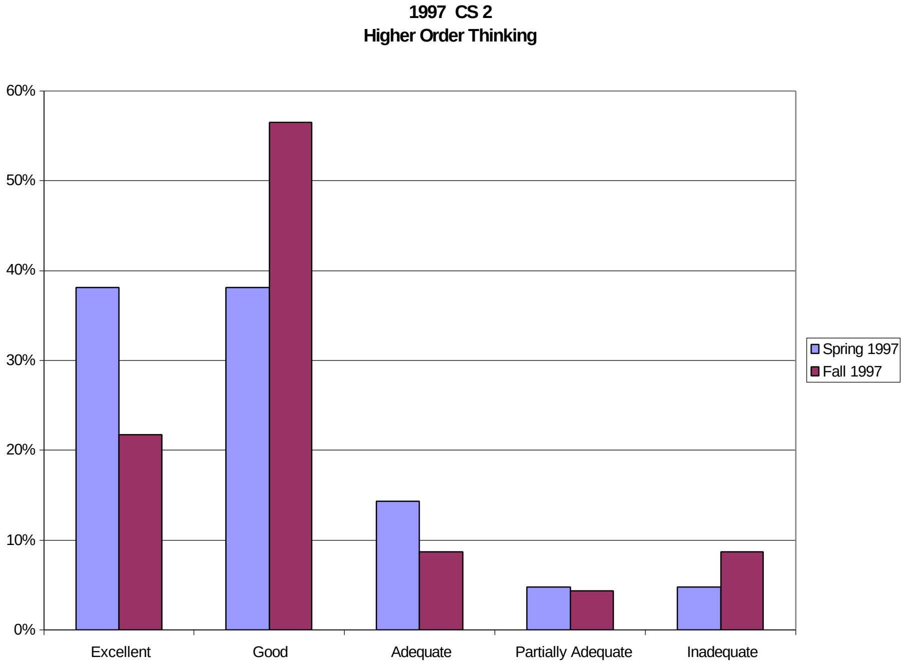


Figure 2: Higher Order Thinking

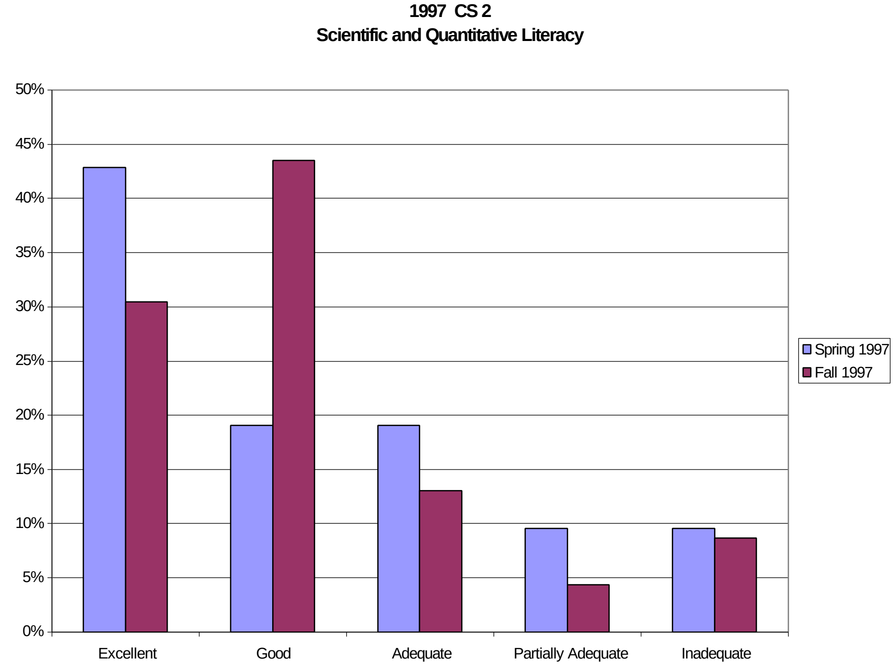


Figure 3: Scientific and Quantitative Literacy

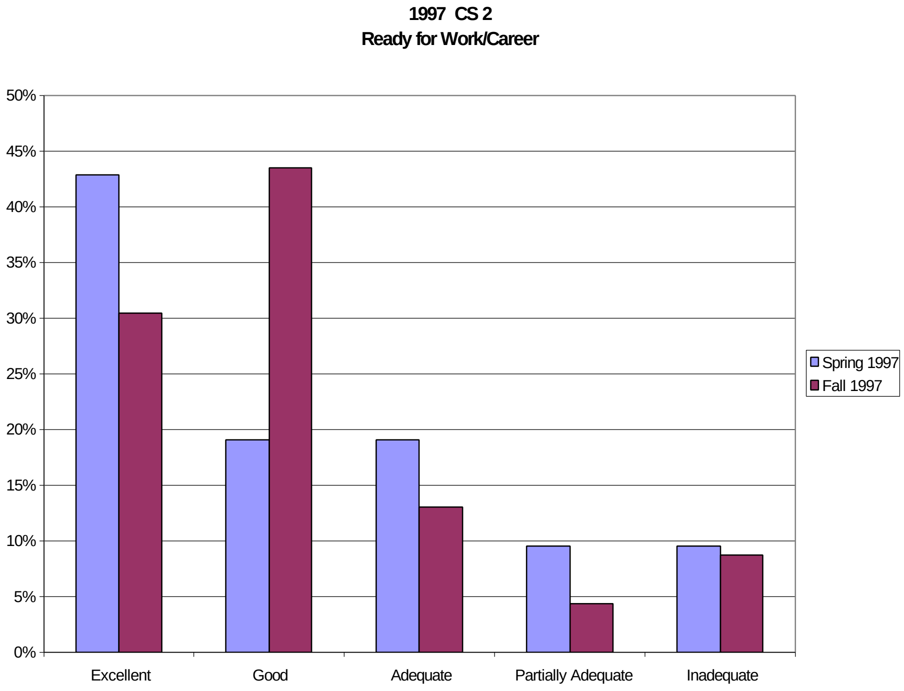


Figure 4: Ready for Work and Career

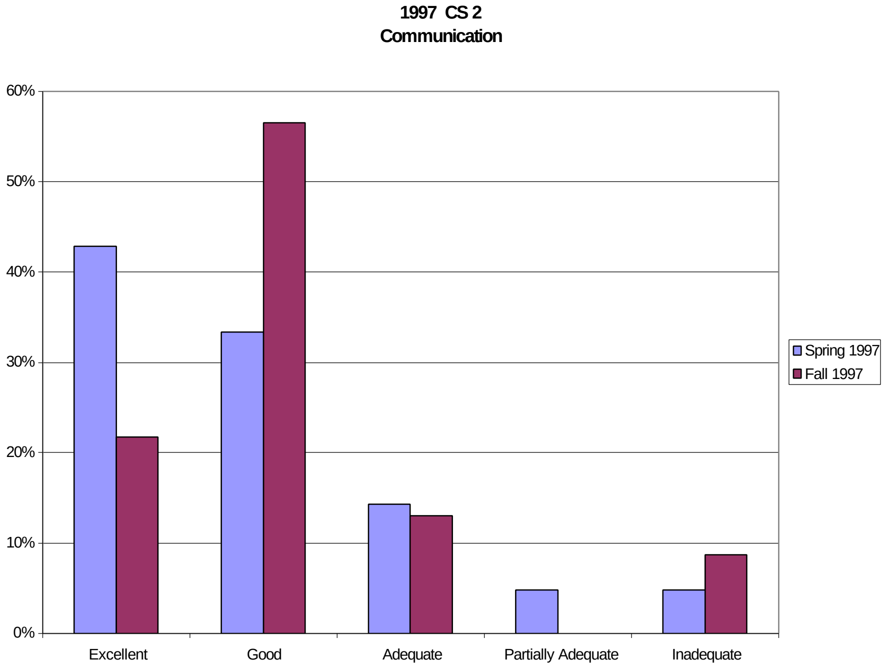


Figure 5: Communication

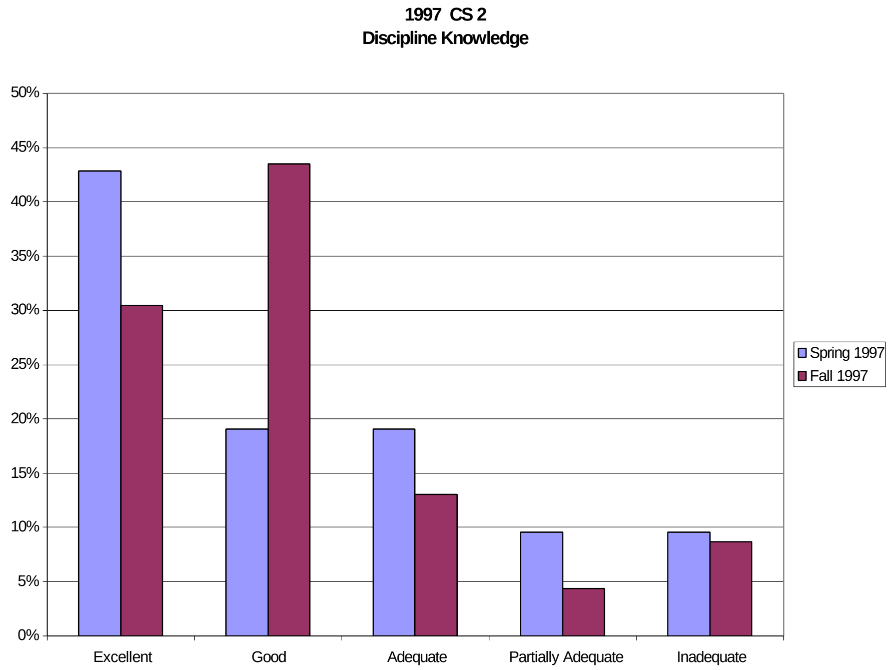


Figure 6: Discipline Knowledge

Spring 1997	Excellent	Good	Adequate	Partially Adequate	Inadequate	Total
Preparedness	16	0	2	1	2	21
Higher Order Thinking	8	8	3	1	1	21
Scientific/Quant Literacy	9	4	4	2	2	21
Ready for Work/Career	9	4	4	2	2	21
Communication	9	7	3	1	1	21
Discipline	9	4	4	2	2	21

Table 1: Spring 1997 Quality Factors

Fall 1997	Excellent	Good	Adequate	Partially Adequate	Inadequate	Total
Prepared	15	5	1	0	2	23
Higher Order Thinking	5	13	2	1	2	23
Scientific/Quant Literacy	7	10	3	1	2	23
Ready for Work/Career	7	10	3	1	2	23
Communication	5	13	3	0	2	23
Discipline	7	10	3	1	2	23

Table 2: Fall 1997 Quality Factors

	Spring 1997	Fall 1997
Prepared for college (Q1)	85.7%	91.3%
Higher order thinking (Q2)	90.5%	87.0%
Quantitative and scientific literacy (Q4), readiness for work and career (Q6), discipline knowledge (Q9)	90.5%	91.3%
Communications (Q8)	81.0%	87.0%

Table 3: Excellent, good or adequate outcomes

Longitudinal data is more difficult to evaluate. On the surface, it could appear that the Spring 1997 class performed better than the Fall 1997 class, with excellent ratings of 76% (65%) for college preparation, 38% (22%) for higher order thinking, 43% (22%) for communication skills and 43% (30%) for scientific and quantitative literacy, readiness for work and career, and discipline knowledge (Fall ratings in parentheses). However, expanding the criteria to include good and adequate ratings paints a different picture as shown in [Tab. 3]. A larger percentage of students in the Fall section performed at least at an adequate level than those in the Spring quarter for all quality factors except higher order thinking.

Although various explanations are possible (e.g., students were better or not as well prepared by prerequisite classes, there was a stronger bimodal distribution of abilities in the Spring quarter, etc.) with only two quarters of data available further generalization is not advisable. Certainly this data will become more informative as trends emerge over time. And, too, as more data is collected, the additional analysis of prerequisite classes will permit us to hypothesize about cause and effect as well.

Conclusions

A comprehensive mechanism for tracking learning outcomes related to measures of interest to accrediting bodies, legislators, employers, and faculty has been developed and implemented at Metropolitan State University. This mechanism has been used during the 1997 Spring and Fall quarters to collect data. Analysis shows that the CS 2 course successfully attempts to cover the quality factors of preparedness, higher order thinking, scientific and quantitative literacy, readiness for work and career, communication, and discipline knowledge with better than 80% of the students completing the course ranked as Adequate or better in all areas.

Trend data shows mixed results with more students performing at an Excellent level in the Spring quarter, but more students performing at an Adequate or better level in the Fall quarter. Trends will become more apparent as additional data becomes available.

The next step in using this data to improve our programs is to analyze the coverage of the quality factors throughout the computer information systems and computer science curricula and to compare outcomes in sets of related courses over time.

The potential uses for this tool are enormous and exciting. The ability to abstract data, to study the patterns of learning outcomes achieved by our students over time, and to discover trends will be an extraordinarily useful tool as our department continues to assess and enhance our curriculum.

References

[Metro State 1996] 1996-1998 Metropolitan State University Catalog. Minneapolis-St. Paul, MN: Metropolitan State University.

[Astin 1993] Astin, A. W. (1993). Assessment for Excellence. American Council on Education, Series on Higher Education, ORYX Press.

[North Central 1994] Assessment of Student Academic Achievement, Appendix A. Handbook of Accreditation. (1994-1995). North Central Accreditation-Commission on Institutions of Higher Education (draft).

[Metro State 1995] Assessment Plan: Student Academic Achievement. (April, 1995). Minneapolis-St. Paul, MN: Metropolitan State University.

[Merriam and Cunningham 1991] Merriam, S. B. & Cunningham, P. M., Ed., (1991). The Evaluation of Adult and Continuing Education. Handbook of Adult and Continuing Education. Jossey-Bass Pub. 260-272.

[Metro State 1997] Progress Report on the Plan for Assessing Student Academic Achievement at Metropolitan State University. (July 1997). Minneapolis-St. Paul, MN: Metropolitan State University.

[Simosko 1988] Simosko, S. & Associates. (1988). Assessing Learning: A CAEL Handbook for Faculty. Columbia, MD: Council for Adult and Experiential Learning.

Acknowledgements

The author gratefully acknowledges the efforts of the Metro State Assessment Committee members who formulated this assessment approach and made the data available for analysis. Particular thanks go to Susan Rydell, Carol Holmberg, and Jeff Mosner.

Deploying Port Manageable Hubs to Create Virtual LAN's to Improve Network Performance in a General Education Computer Lab

Dennis Guster
Professor, Microcomputer Studies Program
Department of Statistics
St. Cloud State University
St. Cloud, Minnesota
USA
guster@mcs.stcloudstate.edu

Aaron Shilts
MCS/CSCI 169 Lab Manager
St. Cloud State University
St. Cloud, Minnesota
USA
aarons@mcs.stcloudstate.edu

Abstract

In recent years a number of factors have placed burdens on network infrastructure. First, there has been an increase in users, even the most unsophisticated user is now generally connected to the network. Second, the prevalent use of GUI operating systems and the integration of video/graphics into the data stream have increased the volume of bits transmitted. This explosion in network usage has been addressed by vendors of networking components with the offering of a number of high-speed solutions aimed at backbone application or high-end users. These solutions are certainly attractive from purely a performance standpoint, but may not be cost effective for some low-end applications. Furthermore, their integration into some existing network structures may necessitate a completely new design.

Deploying Port Manageable Hubs to Create Virtual LAN's to Improve Network Performance in a General Education Computer Lab

Dennis Guster
Professor, Microcomputer Studies Program
Department of Statistics
St. Cloud State University
St. Cloud, Minnesota
USA
guster@mcs.stcloudstate.edu

Aaron Shilts
MCS/CSCI 169 Lab Manager
St. Cloud State University
St. Cloud, Minnesota
USA
aarons@mcs.stcloudstate.edu

Abstract

In recent years a number of factors have placed burdens on network infrastructure. First, there has been an increase in users, even the most unsophisticated user is now generally connected to the network. Second, the prevalent use of GUI operating systems and the integration of video/graphics into the data stream have increased the volume of bits transmitted. This explosion in network usage has been addressed by vendors of networking components with the offering of a number of high-speed solutions aimed at backbone application or high-end users. These solutions are certainly attractive from purely a performance standpoint, but may not be cost effective for some low-end applications. Furthermore, their integration into some existing network structures may necessitate a completely new design.

Port manageable hubs offer a solution for providing low-end users better performance and easily integrate into existing ethernet 10BASET networks. These units allow computers to be organized into work groups which logically can be viewed as a virtual LAN. For example, a lab containing 20 computers could be organized into 4 virtual LAN's each containing 5 computers. This organization may aid in station management and would reduce the number of nodes any given computer in this lab would be competing with for resources. Besides using the ports in these hubs for connectivity from a workstation to the backbone (which would contain servers), it is possible to attach additional pipes (connections) from the backbone to the port manageable hub. Each dedicated connection attached then brings additional bandwidth that can be shared among the virtual LAN configured. Using the prior example if 4 pipes, each a dedicated 10 Mbs, were attached to the hub, each virtual LAN could be linked to its own incoming connection resulting in 5 computers sharing 10 Mbs. This can be a significant improvement over the traditional hub solution. In the example previously sited, all 20 computers would share 10 Mbs if connected to a traditional hub.

Therefore, the purpose of this paper will be to present, through a case study approach, the methodology and advantages of integrating port manageable hubs into general education computer labs. Specifically the process of upgrading 2 10BASE2 segments each containing 30 workstations will be discussed. Drawings will be provided that illustrate lab connectivity prior to and after implementation. Furthermore, the whole process will be presented stressing two concepts: ease of integration and increasing performance in a cost-effective manner.

Need/Rationale

Even in general education computer classes there is a need to provide a reasonably high-speed network connection. These classes often use Windows 95 and web browsers coupled with protocols such as IPX that at least sporadically make substantial demands on network resources. Furthermore, these classes usually enjoy substantial enrollments which necessitate the need for large numbers of supporting workstations and a high-end server attached to a high-speed backbone. [1] [Parnell, 1996]. From the cost perspective, providing switched high-speed networking to the workgroup is still more expensive than utilizing the hub solution [2] [Conover 1996]. The same is true of the integration of 100 Mbs technology to the workgroup level, it has become more competitive but still not cheaper than the traditional hub solution [3] [Guster & Holden 1997]. In fact, a recent survey in *LAN Times* revealed that sales (in total ports) of 10 Mbs hubs were about five times greater than 10 Mbs switches and about ten times greater than 100 Mbs switches or hubs [4] [Stevens 1998]. The article further states that the hub still offered an attractive low-cost solution for many low-end applications such as email, word processing and spreadsheets.

However, as previously stated, many general education classes have transcended beyond the basic operating systems, applications and protocols and may require more bandwidth than a port on a hub can provide. Furthermore, although more bandwidth may be needed, a port on 10 Mbs switch or 100 Mbs technology may be out of the question from either a design or cost perspective.

In some cases upgrading large numbers of workstations to high-speed technology could have detrimental consequences to network performance. For example, the bottleneck might simply be moved from the workgroup to the backbone. An even worse scenario would involve no network bottlenecks but an overwhelmed server that would never be able to keep pace with the rapidly arriving requests. Therefore, there is a need for a compromise solution that would provide bandwidth greater than the 1/24 of 10 Mbs common in hubs but less than the dedicated 10 Mbs port on a switch or 100 Mbs technology.

The Port/Switch Hub

The portswitch can be viewed as sort of a hybrid, part hub and part switch. Typically they are deployed in 24-port units like hubs. The big difference lies in the bandwidth made available per port. In a hub configuration (10BASET) all ports on the hub or hubstack are sharing a 10 Mbs pipe, whereas a port/switch allows multiple 10 Mbs pipes to be brought into the hub or hub stack.

For example, in the equipment used in the case study, it is possible to dedicate 8 ports to incoming lines thus yielding 80 Mbs of bandwidth to be shared among the hub/hub stack [5] [Transition Networks 1996]. To make this concept work it is necessary to create virtual networks. In other words, groups of ports need to be assigned to each incoming line. If 40 workstations needed connectivity, they could be configured into 8 virtual networks. Each virtual network would require 6 ports, 5 for the workstations and 1 for the incoming line. Therefore, the total setup would require 48 ports, 40 for the workstations and 8 for the incoming lines. In terms of bandwidth, each virtual network of 5 workstations is sharing 10 Mbs. This is a great improvement over a traditional hub solution where all 40 workstations would be sharing 10 Mbs, but not as good as a switch in which each port gets a dedicated 10Mbs pipe.

In addition to increasing the available bandwidth beyond traditional hubs, the virtual network capability of the port/switch can help reduce the size of the collision domain. In the example previously sited, 40 workstations are connected together by a hubstack resulting in a collision domain wherein 40 devices are contending for access to the media. In other words, the 40 workstations are sharing a single channel and using a type of time division multiplexing to share the media (CSMA/CD algorithm). Generally speaking, as the number of devices per shared net increases, the probability of collision increases as well. When collisions occur, the line jams and resets which takes away from the actual throughput.

Configuring virtual networks not only makes it easier to increase bandwidth, but often provides a means of reducing collision domains. Once again using the prior example, the number of devices per shared net is reduced from 40 to 5 thus reducing the likelihood of collisions.

The port switch also offers benefits from a user management perspective. With this setup, it is easy to move a group of users from one network segment to another. Or, if a user transfers to a different department, it is easy to delete him/her from his/her current virtual network and add him/her to the new one. This is all possible through software manipulation that uses SNMP or RMON technology.

From a security perspective, the port switch offers the ability to isolate traffic. In the prior example, all 40 workstations shared the same segment and could intercept each other's traffic. Using the port switch to create 8 virtual but physically separate networks reduces the group size that may intercept each other's messages to 5, certainly a more manageable figure.

Design Structure

The case study herein illustrated is based on the upgrade of a general education computer laboratory of approximately 100 workstations. About two-thirds of it was still wired (in two networks) with 10BASE2 coax cable. The other one-third had been upgraded and wired to 10BASET network in a shared (hub) arrangement. Because students typically needed minimum bandwidth for the classes this lab supported, it was given a low priority for upgrading even though performance occasionally decayed to an unacceptable level. Introducing Windows 95 and the extensive use of internet browsers in the courses necessitated an immediate solution. Funding and design limitations precluding all switched ports or shared 100 Mbs technology so, therefore, the port switch solution offered the most cost-effective solution.

Fig. 1 illustrates the state of the network prior to the upgrade. The area affected is enclosed in a dotted-line border. Within this border are two 30-node 10BASE2 networks. Each of these networks are connected to a port on the central 10BASET switch within the microcomputer studies (MCS) equipment room. Therefore, in each case 30 workstations are sharing a 10 Mbs feed.

The drawing further shows that in prior upgrades shared 100 Mbs technology had been introduced into the MCS domain. To take advantage of that existing technology, improve access to 8 production servers, and provide adequate incoming feeds to the port switch stack, the central switching point of the domain needed to be upgraded to 100 Mbs.

As a pilot study, port manageable (segmentable) hubs had already been installed to support the rest of the lab. Fig. 1 shows that there are 4 lines coming into the stack (from a 10BASET switch), providing it with an aggregate of 40 Mbs.

Fig. 2 illustrates the components and topology used to upgrade the 10BASE2 nets. The central point of connectivity within the domain is the production MCS 100 Mbs switch. This 16-point switch provides connectivity to servers and other switches within the network domain. Three of those 16 ports have been allocated to support the general education lab. Each of those ports is connected to a 10/100 switch (switch master). The switch master then has a 100 Mbs pipe incoming and distributes that bandwidth through five 10BASET ports to the port switch (stackmaster). Therefore, each stackmaster stack has 50 Mbs of bandwidth available. That bandwidth is distributed in 10 Mbs increments to 5 virtual nets per stack. Each virtual net consists of 3 to 4 workstations, and open ports are available on each for future expansion of both incoming and outgoing lines.

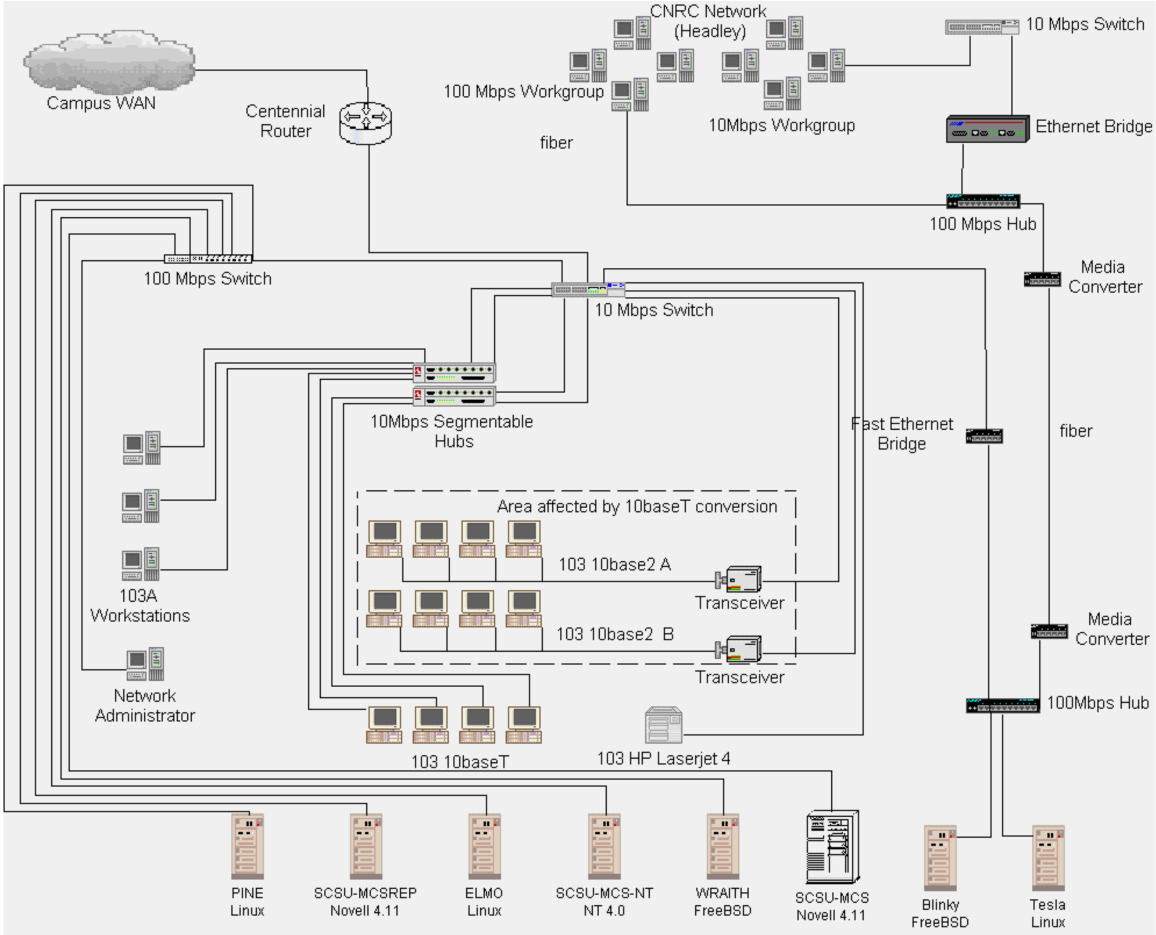


Figure 1: MicroComputer Studies Local Area Network
(with10baseT conversion affect)

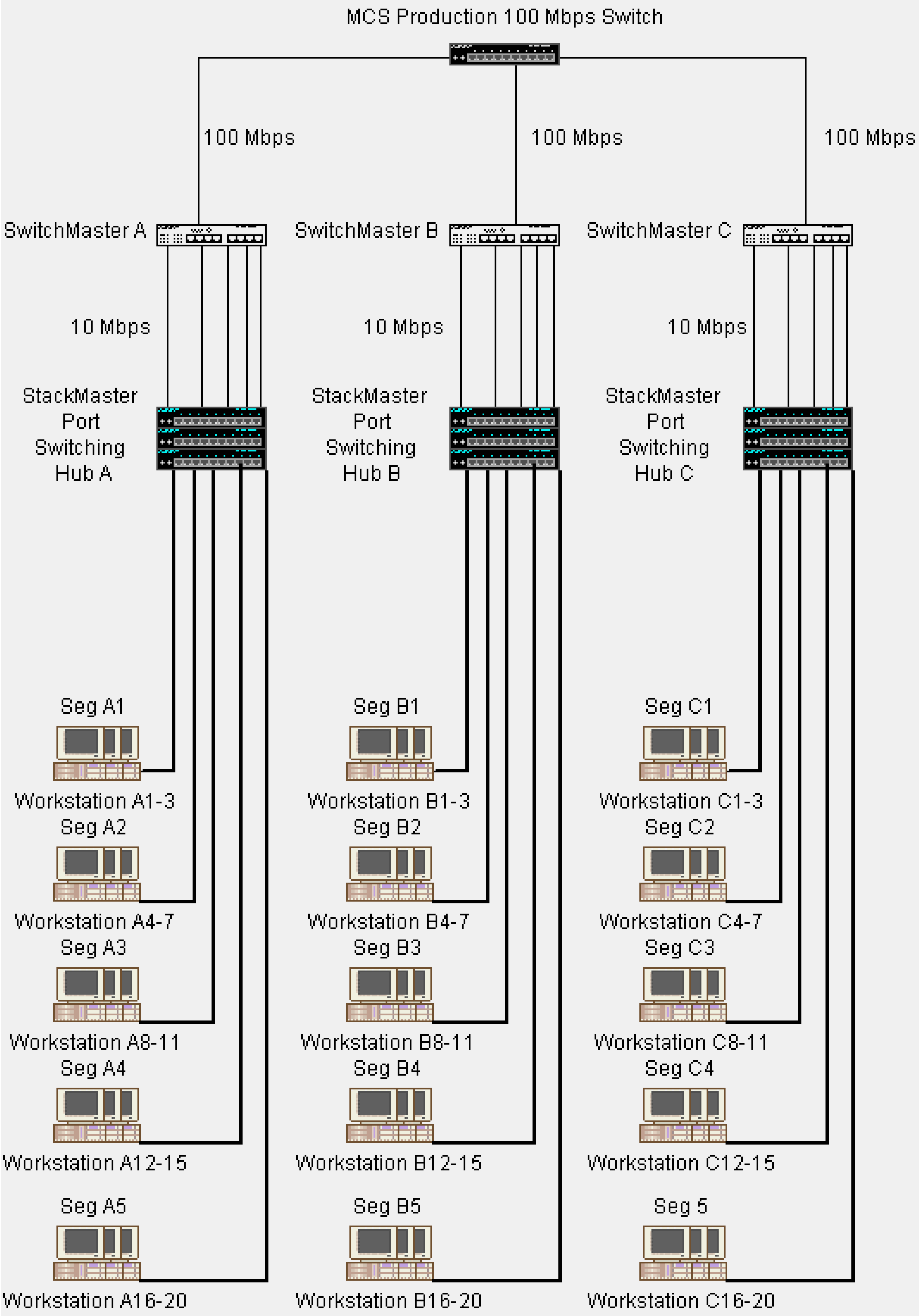


Figure 2: 10base2 to 10baseT Conversion Proposal

Each stackmaster stack consists of a managed port switching hub, an expansion unit, and a redundant power supply. In fact, each stack can be upgraded to accommodate up to 190 ports.

Programming the Port Switch

One of the advantages of traditional hubs is their plug and play capability (assuming a dumb hub). The port switches do in fact require configuration. However, it is a relatively simple process once the design decisions depicted in Fig. 2 are made. Once those decision are made, each of the ports on the Transition Networks Stackmaster Pro hubs can be assigned to individual LAN segments. Programming the ports is extremely simple. There are three methods to assign ports to LAN segments: through the base unit joystick, an attached terminal emulator, or SNMP. Each method is simple, but for the sake of brevity, only the base unit joystick and a terminal emulator will be covered.

Base Unit Joystick

1. Select mode A to put the hub in port assignment mode.
2. Move the joystick right to the unit position.
3. Select the unit (1-8) where the port resides. Since these are stackable hubs, you can have up to 8 slave units.
4. Move the joystick right to the port position.
5. Scroll up until you find the desired port.
6. Move the joystick right to the segment position.
7. Change this to the segment that you would like to assign this port.
8. Push the enter button on the hub.
9. Mode B can be used to view what ports are on any given assignment.

Attached Terminal Emulator

1. Connect a cable to your terminal emulator and the hub.
2. Type ‘su’ to enter superuser mode.
3. Enter the password.
4. At the # prompt use the following syntax to program any segment: *set ptos –u*<unit#> *–p*<port #>-s<segment #>
5. To verify the port assignment that was made, type the following *display ptos –u,*<unit#>-*p*<port #>

In addition to providing for segment configuration SNMP/RMON packages can be used for traditional management functions as well because the master hub in the stack is an SNMP agent.

Integrating into the Existing Wiring Scheme

Integrating this design solution into the existing wiring scheme was relatively simplistic on the incoming side. Three CAT 5 connections were run from the equipment room into the lab and linked to each of the three 10/100 switches. The wiring of the workstations was more complex, however. The existing 10BASE2 coax cable had to be replaced with twisted pair which was run back to a lockable cabinet placed in the back of the lab. Fig. 3 illustrates how the workstations were wired to the patch panel in the cabinet. In the prior shared 10BASE2 bus arrangement, reliability and diagnostics were an issue because if any cable on the bus scheme failed, usually multiple (or all) computers on that net failed as well. The new 10BASET solution eliminates that problem because each workstation is directly wired to the port switch in a star arrangement. Furthermore, in planning for this project, the cost of wire and wiring components was grossly underestimated. It turned out that about 33 percent beyond the cost of the switches and hubs needed to be allocated for wiring.

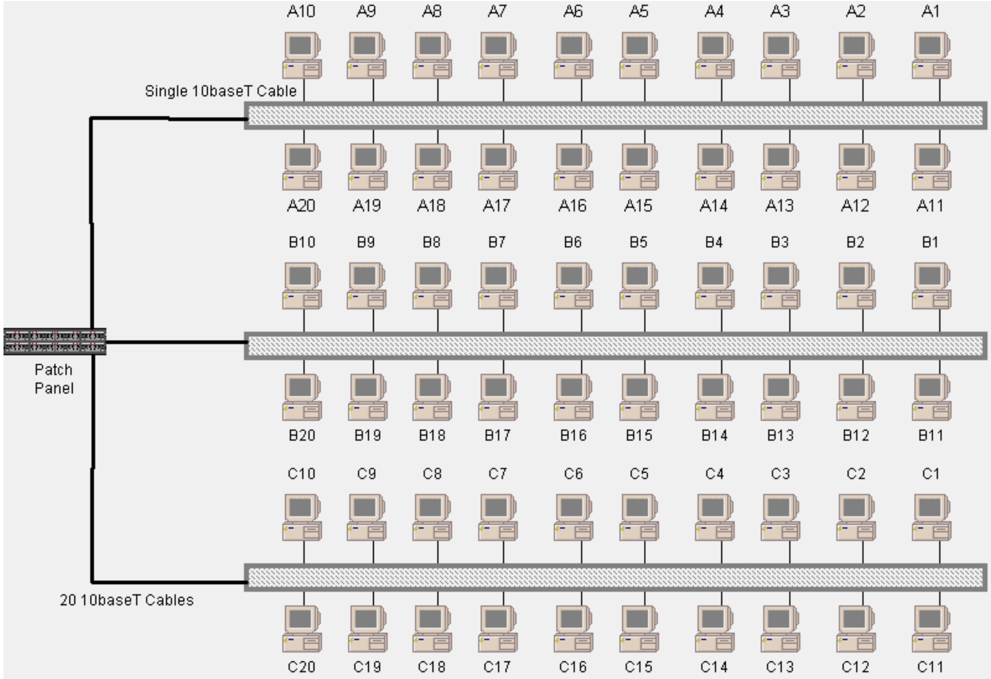


Figure 3: 10baseT Installation

Summary/Conclusions

The solution described in this case study has realized a number of advantages beyond a classical hub solution. First, performance is better. Instead of 24 workstations sharing 10 Mbs, there are no more than 4 workstations sharing 10 Mbs. Furthermore, there will be less congestion back to the domains central connectivity point due to the use of 3 separate 100 Mbs pipes. Also, the collision domains have been reduced from 24 to a maximum of 4 which should significantly reduce jam time.

Second, security was enhanced due to segmentation of workstations into virtual networks. A station put into promiscuous mode will at most only see the packets of three other devices. So, therefore, not only the scope of the threat is reduced, but the probability of determining which station is undertaking the unauthorized listening is increased.

Third, although the port switches were slightly more difficult to program than a traditional hub, they offer more flexibility in dealing with moving workstations to other nets if, for example, a person is transferred within the company. Furthermore, additional bandwidth could easily be allocated by bringing in more lines and reducing the size of a given virtual network.

Last, because packet delivery is based on physical address resolution to a segment, a port switch arrangement has a minimal effect on OSI layer 3 network addressing schemes. In the case study, the workstations simply maintained their existing IP addresses (although they were dynamically assigned from a pool for a given session). In fact, the whole domain of over 200 workstations had previously been reconfigured with bridges so that all production devices resided on a single class C internet license [6] [Guster & Holden 1997].

Based on the results of successfully implementing this case study, it appears that port switches offer a cost-effective compromise between traditional hubs and switches. Given today's bandwidth-hungry applications, contending for 10 Mbs with 3 other stations is certainly preferable to contending with 23 [7][Shell 1996]. Furthermore, the added security, reduction in the size of the collision domain and the flexibility of reconfiguration provide subsequent rationale for selecting the port switch solution.

References

[1] [Parnell 1996] Parnell, T. (1996). The wild wired west. LAN Times, 13:19, 55-64.

[2] [Conover 1996] Conover, J. (1996). Full-duplex 25 Mbs ATM NICS bring low-cost connectivity to the desktop. Network Computing, 7:15, 102-110.

[3] [Guster & Holden 1997] Guster, D. & Holden, M. (1997). Integrating 100 Mbs Technology to the Workgroup Via 100 Mbs Bridges. Proceedings of the 30th Small College Computing Symposium, April 17-19, 1997, University of Wisconsin-Parkside, Kenosha, Wisconsin, 73-80.

[4] [Stevens 1998] Stevens, L. (1998). The right tool for the job. LAN Times, 15:2, 47-48.

[5] [Transition Networks 1996] Transition Networks (1996). Stackmaster Pro User's Guide. Minneapolis, MN: Transition Networks.

[6] [Guster & Holden 1997] Guster, D. & Holden, M. (1997). Integrating 100 Mbs Technology to the Workgroup Via 100 Mbs Bridges. Proceedings of the 30th Small College Computing Symposium, April 17-19, 1997, University of Wisconsin-Parkside, Kenosha, Wisconsin, 73-80.

[7] [Shell 1996] Shell, M. (1996). Planning for the future. LAN Times, 13:20, 53-56.

Acknowledgements

This project was undertaken with the cooperation of Transition Networks, Inc., a corporate partner of the Computer Network Research Center at St. Cloud State University.

NetApp - A Client/Server Application Builder

Hossein Hakimzadeh

Steve Hartman

Department of Mathematics and Computer Science
Indiana University South Bend
1700 Mishawaka Ave.
South Bend, IN 46634
USA
hhakimza@iusb.edu

Abstract

A new era of computing is upon us, one which has moved away from centralized control, and toward a distributed model. Distributed processing yields many technical, economical and societal advantages. Among these are scalability, autonomy, resource sharing, cost efficiency, speed, reliability, flexibility of access and graceful degradation. Considering these benefits, one might echo the claim by some that “the network is the computer”. Putting the excitement aside, there is a serious gap in what the industry expects of educators in regard to training students that fully understand and appreciate the virtues of network application development and what the universities are delivering. Our goal in this paper is to identify the core skills and knowledge necessary to implement networked applications. We accomplish this goal by first outlining the resources available for network application development. Later, we present the design and implementation of the NetApp Class. The NetApp is a C++ class which encapsulates the Berkeley socket library. Our goal is to understand the nature of network applications through the development of a flexible and user friendly C++ class for network and client/server computing.

1 - Introduction

The virtues of distributed and client/server computing have not gone unnoticed, certainly not by employers. Today there is great demand for professionals who understand and appreciate the benefits and complexities of network application development. Unfortunately, the traditional undergraduate programs in computer science have not met the challenge of preparing the students for these demands. As a result, in the past few years NSF has funded a number of pilot programs which have focused on bringing this technology to undergraduate programs[1].

Our goal in this paper is two fold. Initially, we identify and survey the resources available for network application development. Later, we present the design and implementation of the NetApp Class. The NetApp is a C++ class which encapsulates the Berkeley socket library functions. The NetApp enables the application programmer to easily develop network applications by simply allowing the programmer to view the network as just another object.

We begin this paper with the assumption that the reader is familiar with the basic network concepts and terminology, such as physical communication media, network topology, protocols, and the basic networking models such as OSI and TCP/IP. A typical course in computer networks provides excellent

[1] NSF Award Abstracts can be found at http://www.nsf.gov

coverage in these topics. Therefore, we first turn our focus toward examining the resources available for network programming and later describe the design model and implementation of the NetApp class. We conclude by making some pedagogical notes as to how the NetApp class library can provide the student with basic tools and understanding necessary to build real world distributed applications.

2 - Resources For Network Programming

In this section we will provide an introduction to BSD's *Berkeley Socket* and System V's *Transport Layer Interface (TLI)* networking libraries. Our coverage is limited to IPC and networking libraries available for the Unix operating systems. However, Microsoft's and Trumpet's *Winsock* libraries provide equivalent functionality for Windows 3.x, Windows95 and NT operating systems. Interested readers are referred to [Bonner95], [Dumas95], [Winsock95] and [Quinn & Shute95] for additional details.

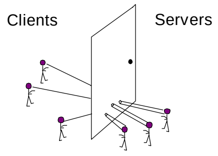
2.1 - Berkeley Socket Library

The Berkeley Socket library is an application program interface (API) that reside within the application layer of TCP/IP and serve as the path through which user applications access lower layer network protocols. Berkeley Sockets were originally included in the 4.1cBSD release for VAX around 1982. The functions included herein are extracted from the 1986 release 4.3BSD for VAX.

The concept of a socket might be visualized by thinking of it as a hole in a door. The server simply waits with its ear to the hole and listens for requests. Actually the server goes into a light sleep. When a client wants to communicate with the server, it speaks directly through the hole (socket). When the server "hears" the client it is awakened and it attempts to process the request.

In a typical client/server application, the above concept is taken further by imagining the door having many holes, each of which is listened to by an identical copy of the original server (see fig. 1). In this model the client must know which hole leads to the appropriate server that is capable of processing the type of request it is making.

Figure 1: Client/Server Model



BSD 4.3 sockets support three types of communication protocols; *Unix domain*, *Internet domain* and *Xerox NS domain.* These protocol domains are referred to as protocol families. The Internet family is the most widely used and will be the only protocol addressed herein.

When a user application requires a network session, it must perform a series of system calls. First it must create a socket and associate the socket with the host's address and the desired port. It can then perform network communication and finally close its socket upon completion of the session. These calls to the operating system are procedures defined within the Berkeley Socket Library.

Many network system calls require a reference to a structure containing a socket address. The definition of this structure is in file <sys/socket.h> and is as follows;

```
struct sockaddr {
  u_shortsa_family;          // address of family : AF_xxx value
  char            sa_data[14];       // up to 14 bytes of protocol-specific address
};
```

This structure can be used by all address families. Because the different address families use different methods to extract the host address and network address from the sa_data field of the structure, the type of address family used determines how the address is interpreted. This necessitates each family having its own storage structures for addressing information. The Internet family uses addressing structures that are defined within the <netinet/in.h> file. The definition is as follows:

```
struct in_addr {
   u_long         s_addr;            // 32-bit netid/hostid network byte ordered
};
struct sockaddr_in {
   short   sin_family;               // AF_INET
   u_short        sin_port;// 16-bit port number
   in_addr        sin_addr;          // 32-bit netid/hostid
   sin_zero[8];                      // not used
};
```

The above structure is Internet address family-specific. Because the system calls used to create and use sockets must support all the addressing families, these structures are cast into the generic sockaddr structure shown previously and passed along with their size to the system functions.

In order to achieve network communication an association or connection between the peer applications must be established. This association consists of necessary information collected by the peer processes about each other in order to route and correctly interpret the data. The following information must be provided in order for the end-users to establish an association [Stevens90].;

protocol	local address	local process	foreign address	foreign process

Table 1: Information needed to establish an association between two peer process.

The *protocol* is an agreement which determines the type of protocol (TCP[2], UDP[3], ICMP[4], IP[5], etc.) shall be used for the session. The *local* and *foreign addresses* are similar to the destination and return addresses used in the postal mail scenario. The *local* and *foreign processes* are the identification numbers of the specific server and client applications involved in the connection.

Figure 2 demonstrates the general sequence of function calls that establish a simple connection between two network processes. An overview of a connection-oriented communication request scenario could be as follows. First the server must create a socket. When a socket is created it is customized to meet the requirements of the application that will use it. The application must provide the operating system with three pieces of information so that it can create the desired type of socket, the address family and the communication type. The socket system call is:

int socket (int family, int type, int protocol);

where *family* refers to the protocol family (Internet, Unix or XNS), *type* refers to the communication type (stream for connection-oriented and datagram for connectionless), and *protocol* refers to an enumerated value specifying a specific TCP/IP protocol.

[2] Transmission Control Protocol (connection oriented protocol)

[3] User Datagram Protocol (connectionless protocol)

[4] Internet Control Message Protocol (low level protocol used mainly to transmit information needed to control IP traffic. Ping is implemented using this protocol.)

[5] Internet Protocol (low lever protocol which receives packets from the physical layer and passes upward. Conversely, it transmit packets received from upper levels to the lower levels.

For example, to create a server application with a connection-oriented socket that would use Internet protocols the socket system call would look like:

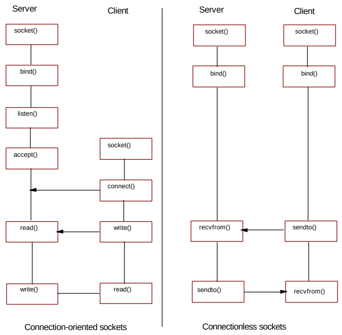


Figure 2: Server and client network system calls [Stevens 90].

socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

The AF_INET parameter requests the Internet Address Family. Alternatively, we could specify AF_UNIX to use Unix domain or AF_NS for XNS protocols. The SOCK_STREAM parameter specifies connection-oriented communications (TCP). A SOCK_DGRAM value specifies connectionless communications (UDP).
The type of *protocol* to be used is implicitly made by the value of the first two parameters. In the case of the call above, the type of protocol would be IPPROTO_TCP (IP protocol - TCP). The third parameter is for user applications that require a specific protocol to be used. Normally this parameter is set to zero which is the enumerated value for IP.

Creating a socket fulfills the initial requirement to establish an association by specifying the protocol. Upon successful creation of a socket, the operating system returns a socket descriptor. The user application then requests that the operating system bind together (create an association with) the socket descriptor, the server's IP address and the server's port number. This is accomplished by using the *bind()* system call. The application stores the addressing information in a address structure which it passes to the bind function. The bind function prototype is:

```
int bind(int sockfd,  sockaddr  *server_addr,  int addrlen);
```

where *sockfd* is the socket descriptor, sockaddr is a structure holding the port and IP address of the server and the *addrlen* is the number of storage bytes used by the *sockaddr* structure.

Once the address information has been bound to the socket descriptor the local address and local process requirements of the association are fulfilled. The next step is predicated upon the server's communication type. If it has created a *connectionless* socket, it waits until a message arrives. Once a message is detected the server application can read it directly from the socket by using the standard I/O *read()* function or a network I/O function such as *recvfrom().*

```
int read(int sockfd, char *buff, nbytes);
int recvfrom       (int sockfd, char *buff,  int nbytes, int flags, sockaddr *from,  int
                   *addrlen);
```

The parameter **buff* is a character buffer that stores the data read from the socket. *Nbytes* tells recvfrom how many bytes to read from the socket and *from* is the addressing structure that will hold the client's address.

If the socket is *connection-oriented* the server must perform additional steps before the socket is ready to receive massages from the network. The server must call the *listen()* system call which establishes a queue to store incoming requests that arrive while the server is processing a previous message. The listen function looks like:

```
int listen(int sockfd,  int backlog);
```

where *backlog* is the maximum number of requests queued at any one time.

After the return from the *listen()* function, the connection-oriented server immediately calls the *accept()* system call. During this function the server blocks for a message to arrive at the socket. When such a message arrives, accept will create and return a new socket descriptor. Depending on the type of server, concurrent or iterative, the server will either *fork()* an exact replica of itself (a child process) to respond to the message or it will respond to the message by itself. In the former case the child server process performs connection-oriented communication with the client through the new socket, thus freeing the parent server and the original socket to receive additional messages from other clients.

During the very brief time the server is making the *fork()* call, any messages arriving at the socket are immediately stored in the queue created in the *listen()* system call. When the server returns from the *fork* , it checks the queue first and processes any messages that might be in it using the same method previously described (see Fig. 3). When the queue is empty it again makes the accept call. The accept function prototype looks like:

```
int accept(int sockfd, sockaddr *peer, int *addrlen);
```

The *peer* parameter of the accept call is a structure similar to the *from* parameter in the recvfrom call. This information includes the client's IP address and data port. With the addition of this information the association is complete and the communication session can begin.

protocol	local address	local process	foreign address	foreign process
socket() TCP	bind() 149.161.9.19	bind() 6010	accept(),connect() 149.161.10.2	accept(),connect() 6011

Table 1: Commands needed to establish an association between two peer process.

The procedure used by a client to create a socket is identical to that used by the server. In order to obtain the desired server's address, the client makes a request to the domain name server using the *gethostbyname()* function. The function returns a pointer to a structure that includes information about the server (IP address, alias names, etc.). The client can also obtain information about specific data ports used by the server by making a *getservbyname()* function call.

If the client desires communication with a connectionless server it binds the descriptor of the socket it has created with its host's address using the *bind()* function. The addressing information is included within the header of the datagram which it writes to the socket using the *sendto()* function.

*int sendto (int sockfd, char *buff, int nbytes, int flags, sockaddr *to, int *addrlen);*

If the client wishes connection-oriented communication, it does not bind its address, rather it calls the *connect()* function. This function establishes the connection between end-users.

*int connect(int sockfd, sockaddr *servaddr, int addrlen);*

The client completes a sockaddr structure with the server host's IP address and desired port number. The *connect* function does not return until the connection between the end users is established or if an error results.

During the connection-oriented session, the end users exchange data by applying the standard I/O functions *read()* and *write()* to their respective sockets.

*int write (int sockfd, char *buff, int nbytes);*

A connection-oriented session is terminated when either of the end users removes (closes) its socket. Note that this is similar to closing a file in file I/O.

Figure 3: Connection-oriented server and child processes.

int close(int sockfd);

2.2 - Transport Layer Interface (Tli)

TLI is a communication library which was first introduced with System V Rel. 3.0 in 1986 [stev90]. TLI provided the programmers with an interface to the transport layer of the OSI model. TLI uses the notion of the *transport endpoint* (processes which seek to communicate) and *transport provider* (set of communication routines). The transport providers are responsible for providing communication support for various protocols such as TCP/IP, XNS and SNA. Therefore, a given system may have many transport providers. TLI provides the interface between the transport endpoints and transport providers. See figure 4.

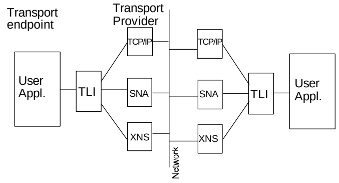
The TLI library is more similar to Berkeley Sockets than it is different. Therefore, in the following table we will provide a comparison between the two instead of a function by function description.

Berkeley Sockets	**Transport Layer Interface (TLI)**	**Comparison**
socket()	int t_open(char * pathname, int oflag, t_info *info)	Defines the protocol such as dev/tcp, /dev/udp or /dev/ip as well as the type of connection such as connection oriented or connectionless. t_open returns a file descriptor which will used by other functions.
bind(), listen()	int t_bind(int fd, t_bind *request, t_bind *return)	Takes the transport endpoint returned by t_open and assigns an address to it. For connection oriented servers, it can also indicates the maximum number of outstanding connection. Which is done by listen() under Berkeley Sockets.
NA	t_alloc(), t_free()	t_alloc() and t_free are used to allocate and free the structures necessary in other functions. No equivalent function is provided under Berkeley Sockets.
See accept()	int t_listen(int fd, t_call *call)	t_listen is used by a connection oriented server to wait for a request. The t_call structure contains the protocol address of the calling transport user, as well as any user data send by the client. T_listen appears to be similar to BSD accept() however, it simply listens for requests it does not accept it. The t_accept will perform that task.
See accept	int t_accept(fd, int req_fd, t_call *call)	Once t_listen declares that is request is made, t_accept() can be used to accept the request. A typical concurrent server will fork an new process to connect to the requester's fd.

accept()	t_open()
	t_listen()
	t_accept()
		The BSD accept() system call does the job of t_open, t_listen and t_accept() functions.

connect()	t_connect(int fd, t_call *sendcall, t_call *rcvcall)	t_connect is used by the client and is similar to BSD connect() however, it is able to set and get some information about the connection which BSD must specifically ask for by invoking other functions such as setsockopt() and getpeername().
sent(), recv	t_snd() , t_rcv()	t_snd() , t_rcv() are used by the connection oriented servers and clients.

Figure 4: Transport Layer Interface



sendto(), recvfrom()	t_sndudata() , t_rcvudata()	t_sndudata() , t_rcvudata() are used by the connectionless oriented servers and clients.

close()	t_sndrel(int fd)
	t_rcvrel(int fd)
		Sends an orderly release of a connection.
		Receives and orderly release of a connection.
		Guarantees that any pending data is delivered.

NA	t_snddis(int fd, t_call *call);
	t_rcvdis(int fd, t_discon *discon)
		Send an abortive release of a connection.
		Receive an abortive release of a connection.
		Abortive release does not guarantee the delivery of pending data communication.

NA	t_close(int fd)	closes the transport endpoint but does not terminate the connection. Used in concurrent servers, in which multiple processes are using the same connection and just because one of them wants to close the transport endpoint, it doesn't mean they all want to quit.

Table 3: Comparison of Berkeley Sockets and TLI

As with Berkeley socket library, the TLI provides a number of additional utility functions which are not discussed here. The interested user is referred to [stevens90] and [SunOS90] for further reading.

3 - NetApp Design Model

The goal of this project was to develop a user friendly API which could be used by students in undergraduate classes to develop simple client/server applications. We began our work by reviewing the existing networking libraries available on Unix. Soon we started to focus on BSD's Socket library which was widely available on all the Unix platforms we had access to. These platforms included HP/UX, Sun/OS, Linux and FreeBSD. Our goal was to encapsulate the Berkeley Socket library in a class hierarchy which allows its users to create network applications by simply instantiating a class. Hence, the Net_App class was born. The following code section demonstrates how a simple file transfer application may be implemented using the NetApp class.

```
// Create a connection oriented server to perform file transfer
#define FT_SERVER_TCP_PORT          6010
#define CONNECTIONLESS              0
#define CONNECTION_ORIENTED         1

main()
{
        int DONE = FALSE;
        NetApp myApplication(CONNECTION_ORIENTED, FT_SERVER_TCP_PORT);
        while (!DONE)
        {
          switch( App.GetRequest() )    {
            case SEND:   if( SendFile(App, App.ReturnData()) == 0 )
                      App.Error( ERR_OPEN_STRING, SERVER_ERROR );
                      break;
            case RECEIVE: if( ReceiveFile(App, App.ReturnData()) == 0 )
                      App.Error( ERR_OPEN_STRING, SERVER_ERROR );
                      break;
            case SHUTDOWN: DONE = TRUE;
            default:       App.Error( ERR_PROCESS_STRING, ERROR_CODE );
          }
        }
        App.ShutDown();
}
```

In the code sample the application programmer is able to create a connection oriented file transfer server, by instantiating an object of type NetApp. Then the server can simply wait for a request (SEND, RECEIVE or SHUTDOWN) to arrive. The SendFile() and ReceiveFile() functions are local to the server. The additional layer of abstraction provided by the NetApp class allows an application programmer to be more concerned with their application rather than having to become an expert network programmer.

From the point of view of a student in a computer networking course, it is much easier to first create a simple client/server application that works, and then proceed to peel away the layers of abstraction. From the point of view of students in other courses such as database management systems or an operating systems who plan to develop a client/server database or a distributed filesystem, the NetApp class can easily be used to implement such projects without extensive discussion of networking topics.

.

Fig. 6 - Data Flow through the Net_App Model

The NetApp class library attempts to model the layered protocol concept of the OSI by creating a hierarchy of objects. Each objects resides on its own logical level and performs functions that are transparent to the objects on adjacent levels. In the following section we will provide some details about the implementation of the NetApp class. The complete C++ source code for the NetApp and its related classes are provided on the world wide web. (http://cosmos.iusb.edu/netapp)

4 - Net_App Class Implementation

The Net_App hierarchy consists of four basic elements, at the highest level is the user application, which is furthest removed from the network. At the lowest level is the Berkeley Socket library which is nearest to the network (not shown in the figure). In between are a Net_App object and the Sock object which provide an interface between the user and the socket. Data flow is directed toward the network when the user wishes to send information, downward through the hierarchy from the user through the Net_App object, through the Sock object and ultimately onto the network through the Berkeley Socket. When the user desires to receive data the flow is reversed. (See Fig. 6)

To perform network communication, the user application instantiates the Net_App class. The creation of the Net_App object causes a chain of events that result in the automatic instantiation of a Sock object (which is the exclusive property of the Net_App object) and a Berkeley Socket (which is the exclusive property of the Sock object).

The Net_App object is created by an overloaded constructor. The user must supply information to the constructor through the values of the actual parameter list so it can determine properties and functions of the Net_App object. The basic values which are passed as parameters are the type of communication (connection-orient or connectionless) and the port to which the Berkeley Socket is to be bound. If the base parameter list is used, the Net_App is created for a server application. The server constructor prototype is

net_app (int Communication_Type, u_short Port_No)

and a call to the constructor would look like:

net_app   My_Sever_App(Communication_Type, Server_Port)

If a forth parameter is included specifying a foreign host's name, the Net_App is customized for a client application. The client constructor prototype is

*net_app (int Communication_Type, u_short Port_No, char *Server_Name)*

and a call to the constructor would look like:

net_app   My_Client_App(Communication_Type, Server_Port, Server).

The communication type parameter implicitly determines the protocol to be used (TCP for connection-oriented, UDP for connectionless).

During instantiation of a Net_App object, the Net_App constructor creates an object of type Sock by calling the Sock class constructor. The Sock constructor is also overloaded for client and server applications. The inclusion of a server host's name within the parameter list determines that a client Sock is required. Upon instantiation of the Sock, the Sock constructor requests a Berkeley Socket from the operating system which it customizes and maintains according to the communication and protocol properties established when the Net_App is created. The Sock object is the exclusive property of the Net_App object.

Figure 7: Client/Server Communication Session Initialization and Data Transfer Using Connection-oriented Communication (Client Application is The Sender in This Example).

The basic function of a Sock object is to process data to send to either the network through its Berkeley Sockets or to its parent Net_App object. The Sock object encapsulates data into structures called Packets which are objects of the class *Packet.* Packet objects combine the data with addressing and data formatting information that allows the sender and receiver to not only route the data along the network, but make and respond to requests, keep track of the data transmitted and acknowledge successful communication.

A packet can be of three basic types: request, acknowledgment or data. Acknowledgment packets and request packets contain only header information. *Request* packets objects consist of values relating to type of communication, direction of data transfer and network addressing information.   An *acknowledgment* packet is similar to the request packet and is sent in response to a request or acknowledgment of a successful or unsuccessful data transmission event.

A server application uses the Net_App member function *GetRequest()* to direct its Application's Sock object to monitor its Berkeley Socket which polls the network for messages intended for the server application.

To initiate a network session, a client sends a request to a particular server by using the Net_App function *MakeRequest()* and passes the server's name and an enumerated value associated with the type of request (file transfer from client to server, file transfer from server to client, etc) to the Net_App object. The Net_App object instructs its Sock to create a request packet. The Sock makes a system call to the network database to obtain the local host's network address which is used as a return address for the server. It also makes another system call to the network database to obtain the server host's network address to route the request packet. It adds this addressing information to the packet's header to complete the packet.

Once the request has been received by the server, its Sock strips the addressing information and stores it for future reference. It passes the request to the Net_App which instructs the Sock to create an acknowledgment packet and send it to the client, acknowledging the request. The Sock uses the addressing information it stripped from the request packet to route the acknowledgment packet to the client. The Net_App then sends the request to the server application which determines how to fulfill the request.

The remainder of the session consists of the client and server applications using their Net_App Read( ) and Write() functions according to their duties within the data exchange. The Net_App Write( ) function passes data supplied by the user application to the sock object. The sock object encapsulates the data in a data packet object which it sends to the foreign end user.

If the user application is sending data to the foreign end user, it initially passes the total number of bytes it will send to the Net_App. It then proceeds to send the data to the Net_App. The Net_App stores the data in its buffer and continually compares the total number of bytes to be sent with the running total passed by the user. When the buffer becomes full (the buffer has a predesignated size) or the data has been passed in its entirety, the Net_App object passes the data to its Sock object. The Sock packages it in a data packet and supplies the header information. It then sends the data packet to the foreign end user application.

When the foreign host receives the data packet, its Sock strips the header to analyze the addressing information. The Sock then passes the remaining data to its parent Net_App. The Net_App stores the data in its buffer where the user application can retrieve it.

If the client application specifies UDP protocol the data exchange is one way, from sender to receiver. If TCP protocol is used, the receiver creates a acknowledgment packet object with every data packet it receives and it sends to the foreign host.

Once the data exchange has been completed, the session ends and the client function terminates. The server application calls the GetRequest() function and blocks until a future request arrives.

Conclusion

The NetApp is a C++ class for the development of client/server applications. The primary goal of our project has been to develop a high level abstraction which can aid computer science students with developing simple client/server applications. Example of such applications and projects in an operating systems course can include the development of distributed filesystems, distributed memory management or implementation of remote print spoolers. In a database management course, the NetApp can be used to implement a distributed query processing system. Similarly, NetApp can be used to implement object migration in a object oriented database system. Other such examples are only limited by the imagination of the instructor's. We plan to continue to refine and extend the Net_App class library and continue to incorporate it in our undergraduate curriculum. We feel that NetApp Provides the students with a unique vehicle for understanding the client/server paradigm.

References and Bibliography

[Bonner95] Bonner, P., (1995). Network Programming with Windows Sockets, ISBN: 0-13-230152-0, Prentice Hall, Englewood Cliffs, New Jersey.

[Douba95] Douba, S., (1995). Networking Unix, SAMS publishing.

[Dumas95] Dumas, A., (1995). Programming WinSock, ISBN: 0-672-30594-1, Sams Publishing, Indianapolis, Indiana, 1995.

[Hakimzadeh98] Hakimzadeh, H., (1998), "Intranets", Department of Mathematics and Computer Science, Indiana University South Bend, Technical Report.

[Hartman96] Hartman, S., 1996. Berkeley Sockets and TCP/IP Implementations in Local Area Networks, Honors Project, Department of Mathematics and Computer Science, Indiana University South Bend.

[Quinn & Shute95] Quinn, B., and Shute, D., (1995) Windows Sockets Network Programming, ISBN: 0-201-63372-8, Addison-Wesley Publishing Company, Reading, Massachusetts.

[Stevens90] Stevens, W.R., "Unix Network Programming", Prentice Hall Software Series, 1990.

[Stevens92] Stevens, W.R., (1992). Advanced Programming in the Unix Environment, Addison Wesley, 1992.

[SunOS90] Sun/OS Manual on Network programming.

[Tanenbaum96] Tanenbaum A. S., (1996). Computer Networks, Third Edition, Prentice Hall Software Series.

[Winsock96] "Windows* Sockets 2 - Application Programming Interface, An Interface for Transparent Network Programming Under Microsoft Windows" Revision 2.2.0, May 10, 1996, http://www.trumpet.com.au/

[NetApp96] Hakimzadeh, H., Hartman S., "NetApp - A Client/Server Application Builder, http://cosmos.iusb.edu/netapp

Surfin' The Net: A Personal Finance Class Project

Dr. Mary W. Hallquist, Professor
Department of Family and Nutrition Sciences
Concordia College
901 S. 8th Street
Moorhead, MN 56562
hallquis@gloria.cord.edu

How can one challenge undergraduates to become intelligent consumers in the future while teaching them the value of technology in making consumptive decisions? The question was unsettling.

Research suggests that having successful instruction is dependent on learner involvement. In addition, there is greater retention of subject matter when students have the opportunity to "show" and also "tell" others what they have learned. The idea of technology enabling students to actively engage in the construction of knowledge gave impetus to the personal finance class project.

The process of development is explained as I rewrote and refined the goals of the project, evaluated available software and cyberspace resources, and established a hierarchy of student learning activities. Although there were successes and high levels of student satisfaction, the project has raised many pedagogical questions in relation to the role of technology in assisting students in their pursuit of knowledge.

Surfin' The Net: A Personal Finance Class Project

Dr. Mary W. Hallquist, Professor
Department of Family and Nutrition Sciences
Concordia College
901 South 8th Street
Moorhead, Minnesota 56562
hallquis@gloria.cord.edu

Introduction

How can one challenge undergraduate students to prepare themselves to be intelligent consumers in the future and at the same time teach them how valuable technology can be in assisting them in making wise consumptive decisions? That was the issue that faced me as I started revising a course that I had taught at Concordia College-Moorhead since 1983. In the past few years I have attended several professional conferences and workshops that addressed the issue of technology in the classroom and its value in teaching personal finance. Most of the presenters were competent, confident users of technology who spoke enthusiastically about its potential. In order to keep abreast of current methodologies, I felt pressured to adapt my past teaching techniques and work to better prepare students for the 21st Century.

A Rationale for the Methodology

In order for instruction to be successful, it must engage the learner [Hitch and Youatt 1995]. Learner involvement is likely to increase and learner distraction will decrease when teaching methods focus on learner participation. The ultimate instructional goal is to engage learners in as much mental activity as possible including not only the processing of information but also in the formulation of questions, in the development of creative solutions, and in the analysis and evaluation of decisions made. Stimulating this type of activity can be challenging to even the most experienced professor. Randy Bass (Georgetown University) presented a lecture for the Technology in the Classroom Workshop at Concordia College ([une 16, 1997] where he stated that teaching must be *interactive* which he defined as engagement combined with feedback. He stated further that technology enables active engagement in the construction of knowledge.

Generally, there is a greater retention of subject matter when students use severl senses in carrying out a learning activity [Chamberlain 1992]. Whenever feasible, students should not only hear about the topic under consideration but also see pertinent materials relating to it. Chamberlain goes on to say that retention of subject matter is about the same whether students hear the information or read the material. However, Chamberlain notes that retention of information over time (3 hours to 3 days) is enhanced when the learner can "show" as well as "tell" others what s/he has learned [1992].

The 'Net project that I developed and implemented in class incorporated the learning principles listed above. Learners were engaged in dialog with other students and with me, computers allowed them to pursue additional sources of information and discuss with their group members the appropriateness of the sources. They asked questions of one another and challenged some of the responses.

The Process of Development

Personal financial management is a major portion of a semester long course, Consumer Economics, which is offered each year by the Department of Family and Nutrition Sciences. The course typically attracts approximately 40 students whose majors/minors are, for the most part, in the social sciences. Students possess a wide range of computing skills from "very confident in using computer applications" to "can you help me turn on the computer?" The text, *Economic Issues for Consumers* by Roger L. Miller and Alan D. Stafford, 8th Edition, is a comprehensive presentation of information deemed useful by students as they prepare to face complex and difficult consumer decisions. It is expected that students will apply the information presented in class and in the textbook as they seek solutions to current and/or future consumer situations.

In spite of the fact that I have taught a Consumer Economics class for 19 years, I have continued to wrestle with the long-term student assignment related to the development of a personal or family budget plan. Students have typically been assigned to participate in a group of 2-4 assuming the roles of members of a specific family unit such as a 22 year old single female school teacher, a 22 year old single male accountant, a 23 year old single graduate student employed at a fast food establishment, a 21 year old single parent employed at McDonalds with a preschool age child, a professional married couple with no children yet, a retired couple living on meager pension benefits, or a single elderly person who has never been employed outside the home but has a small pension from his/her deceased spouse. Students have been asked to write short- and long-term goals and develop plans for achieving the goals, estimate annual family income, project annual expenses, and finally analyze the successes and failures of their plans. It has, in the past, been a rather tedious pencil-and-paper type project, the reality of which has been difficult for many of the students to experience.

During the summer of 1997, I was a participant in a *Technology in the Classroom* workshop for faculty teaching at Concordia College - Moorhead. We were challenged as participants to begin to develop technologically appropriate projects and/or assignments that could be implemented into one or more of our regular classes. This seemed the perfect opportunity for me to lay the groundwork for the family budget plan which I had struggled with for several semesters. Although none of the workshop facilitators taught in my discipline, they were very helpful in assisting me as I began to formulate my plans. Because of their experiences with technology as a teaching tool, they were able to answer many of my logistical questions and inspire me to push forward. Since the personal finance project was going to comprise a major portion of the students' grades for the course, I felt that some class time needed to be scheduled in one of the computer labs on campus. Lab time would be devoted to both formal instructions by me as well as to preparation/completion of student homework assignments.

My first thought was to use a professional home budgeting package, *Quicken Financial Suite* developed by Parsons Technology. I was able to purchase the program with a Concordia College Bush Technology Grant for classroom software and to enroll in a local community education class to become proficient in its use. Although it did not meet all of my expectations, I felt that it would be appropriate for use by college students in completing their budget simulations. As I began developing sections of the long-term budget assignment I became more and more confident of my choice and spoke enthusiastically with my colleagues about the virtues of *Quicken.* It is difficult to describe my feelings of dismay and panic during the first week of the semester's classes when I was informed by the college's network coordinator that I would have to purchase a licensing agreement in order to allow all students to have simultaneous access to *Quicken.* With no funding available to support that endeavor, I had no choice but to develop a contingency plan.

It was then that the *Surfin' the Net* project materialized. The project began with an introduction to the Internet, use of selected search engines (including *Dogpile,Magic Search, Yahoo and Lycos*) and retrieval of information from specific state and national Web sites. Next, the students were required to retrieve pertinent consumer information using search engines and evaluate the quality and usefulness of the information they had located. Students were later introduced to a variety of mathematical calculations available on the Net that were helpful to consumers in making purchasing decisions. At that point students assembled in their small groups, selected a family/individual budgeting case study, and began in earnest the process of developing a workable and relevant annual budgeting plan. The remaining computer lab sessions focused on locating and using Web sites for estimating specific consumer purchases such as housing, transportation, clothing, food as well as plans for savings and retirement and use of consumer credit.

A total of seven class sessions (70 minutes each) were scheduled in one of the student computer labs on campus. Assignments that were developed and used are 1) Search Engines: Locating Consumer Information; 2) Evaluating Consumer-Appropriate Information in Cyberspace; 3) Helpful Mathematical Calculations for the Consumer; 4) Introduction to *Money Magazine's* Budget Planner; 5) Preparing a Personal/Family Budget Plan 6) Analyzing consumer Savings and Spending Options; and 7) Testing Your Budget in Virtual Reality. Each succeeding assignment increased in complexity requiring students to perform more tasks on their own and to employ higher level thinking skills in evaluating the information they were accessing on the Internet.

Many Web sites were accessed. Some of the more useful ones were 1) http://www.kiplinger.com which contains information on investing, family matters, yields and rates, business and market news, financial calculations; 2) http://www.consumerworld.org which contains general consumer hints and provides a variety of assistance in analyzing consumer credit opportunities, searching for bargains; 3) http://www.stats.bls.gov the site of the U.S. Bureau of Labor Statistics which posts a variety of economic indicators including the *Consumer Price Index*; 4) http://www.tmn.com/cdf/index.html the Children's Defense fund which was used to estimate costs of raising a family; 5) http://www.womenswire.com/ with information on using and investing your money; 6) http://www.pueblo.gsa.gov for access to Consumer Information Center information, resources, references and brochures; 7) http://www.emich.edu/public/coe/nice/nice.html for assistance in decision making and links to other useful Web sites; and 8) http://www.kbb.com for information on purchasing transportation.; and 9) http://www.ag.state.mn.us/consumer/ which contains considerable information relating to personal finance.

The final project, which was completed at the end of the 15th week of class, was a print out of the budget that each group had developed accompanied by comments which explained all of the budget expenses. In addition, each group prepared a paper that focused on the major problems/concerns/issues that they faced in allocating the finances and the suggestions for alleviating or overcoming the problems.

Evaluative Comments by the Students

At the end of the semester, students were asked to complete a written evaluation of the Internet project. The questions posed were open ended (your instructor would like to know what you feel has been done well in the teaching of the class). Responses were very positive. Following are some of the examples:
"Bugeting assignment was good…," " Good use of technology integrated into the classroom," " I liked the fact that we were able to use the computer lab," "The budgetr project was interesting because you get different thoughts about what should be included," "Internet assignments were interesting," "The class time in the computer room was very helpful," I enjoyed the variety,…computer labs and simulations," "I felt the budget project was a very good indication for what financial difficulties one may run across in real life."

A second question was posed: Your instructor would also like to know what suggestions you have for improving and strengthening the teaching and administration of this class. Students made some insightful comments. "We also thought it may have been more applicable if we had been doing a budget that was more personal instead of someone with children who is collecting social security." The budget project may have been more meaningful if we had done our own budget – more realistic." "On the budgeting assignment, have each person do it on their own because everyone has different values. Allow students to do it for themselves for the future." "Change the budget project – perhaps project our future rather than give us a phony situation we don't really have an interest in and/or clue about." "The budgeting could have been catered to our own situation – so we see what we are or will be facing…"

In summary it appears that the students felt that the project was worthwhile, that they learned from participating in it, and they felt it was appropriate to use computer technology to complete the assignment. However, many of the students felt that the assignment would have been more realistic had they been able to construct a budget for themselves rather than for a fictitious person.

.

Highlights and Pitfalls of the Project

The students became quite involved in the project and seemed to spend a considerable amount of time discussing the specifics of their budgeting situation. There were allocations of expenses that needed to be made but the students often lacked the knowledge regarding amount or frequency. They would then get on the 'Net to search for information. Some tapped into chat rooms or list serves where they were able to pose questions or query others. Groups were also willing to share information with other groups—an indication of the cooperative nature possessed by the students. It was heart warming to me to experience the enthusiasm and joy they expressed when a problem was solved or an answer was found.

Students gained experience in interacting with the computers and became more competent and confident as the weeks went by. It took them less and less time to locate necessary information as their expertise on the Web increased. One student spoke enthusiastically about her weekly telephone calls to her father to tell him about what she had found on the Internet. Impromptu discussions occurred frequently as groups of students discussed the reliability of information they had retrieved. Most of the time they were able to evaluate it quickly and efficiently—with little input from me.

After having reflected on the process and discussing the outcomes with some of my colleagues I still have some concerns about the worthiness of the project. The more than seven class periods spent in the computer laboratory reduced regular classroom time accordingly and as a result I felt that I was not able to cover adequately the usual course content. Was the trade-off worthwhile? Did the students gain enough through the computer activities to offset what they lost in content? Is the paradigm shifting? I'm not sure.

Students were often frustrated during sessions in the computer laboratory since there were seldom enough computers for each student to have his/her own. It was not uncommon to have 2 or 3 non-functioning units leaving only 13 or 14 PCs for 17 or 18 students. Computer breakdown seemed to occur frequently; repairs were made when service personnel were available – not necessarily in time to benefit any of my students.

My lack of experience in using the Internet interfered at times. Students would not be able to locate the proper screen and I might be able to assist them while other times I had no idea of what was happening. There were some students in class whose computing skills were far superior to mine; fortunately, they were usually willing to assist others in need. It was, however, discomforting for me.

There were times when I felt I had completely lost control of the class, that students were completing so many different tasks that I wasn't able to keep up with them. It made me "stretch" as a teacher, to try and view my position as facilitator and not as the bearer of knowledge. The project took great effort to develop and was tedious, to say the least. It was difficult for me to judge how much time needed to be allotted to complete the assigned tasks in the lab—sometimes we finished early and other times we ran late. Locating appropriate and useful Web sites was also difficult. In contrast, evaluating the quality of the sites was relatively easy, perhaps in part because of material I presented in class lectures/discussions or information that students got from reading their textbooks.

In Conclusion

My ideas about teaching have changed somewhat; I am more convinced now that students learn more readily when they are actively involved in the learning process. I feel that the professor's role of "lecturer" is not nearly as effective as his/her role as "facilitator." Learning seems to be enhanced when the teacher has "set the stage" and encourages students to pursue, question, evaluate and ponder the information they are locating themselves. Technology seemed to assist students in their pursuit of knowledge.

If the goal of higher education is to prepare students for life after graduation, then this project was successful. One student summed it up aptly: "Good basic information presented which will be beneficial in 'real life.'" It is too early to comment on any long-term effects.

The Networking Option within the MIS concentration

Stephen Hawk
School of Business and Technology
University of Wisconsin - Parkside
hawks@uwp.edu

Abstract

This paper discusses how our Business program's MIS concentration allows students to gain some knowledge and experience in networking. Currently, there are two Business courses that teach students networking concepts and skills. The paper discusses the knowledge and skills students gain in these courses and provides an evaluation of how successful our approach has been. Finally, possible future directions for networking option within the MIS concentration are discusses.

The Networking Option within the MIS concentration

Stephen Hawk
School of Business and Technology
University of Wisconsin - Parkside
hawks@uwp.edu

Introduction

MIS programs traditionally have emphasized the system development career path. Almost all MIS programs provide courses in programming, database, and systems analysis and design. Relatively few MIS programs offer even one course in networking or data communications. The DPMA and ACM model curricula also have this emphasis. The DPMA model curriculum, for instance, includes no networking courses among the 9 suggested "core" MIS courses. There are two networking courses included in the 16 possible MIS elective courses defined by DPMA.

Our MIS curriculum also emphasizes the system development career track for the majority of students' course work. Our MIS program, however, also provides some opportunity for students to develop knowledge and skills that could help them to obtain a networking position upon graduation. This paper discusses how this is currently done in our Business program's MIS concentration. Also discussed is an evaluation of this option's success and its shortcomings, and some plans for extending this option.

Overview of the MIS Concentration

MIS is one of five concentrations in the Business Department. All Business majors are required to take Business Core courses that emphasizes basic skills (math, quantitative methods, economics, communications) and the functional areas of business for a total of 48-58 credits. There are currently eight MIS courses in the MIS concentration beyond the introductory MIS course that is required of most Business majors. MIS students take seven courses for a total of 21 credits in the concentration. There are six MIS courses required of all MIS students. These courses are C++, Visual Basic & PowerBuilder, Database Systems, System Analysis & Design, Field Project, and Microcomputers & Local Area Networks. We are considering the addition of a second C++ course to the required set of MIS courses. MIS students may then select from one of two electives: AS/400-RPG programming, and Data Communications.

The two courses that have significant networking content are the Microcomputers & LANs course and the Data communication course. These correspond approximately to the DPMA electives "Implementation of PC Networking Systems" and "Telecommunications" respectively. The following sections describe these courses.

Microcomputers & LANs (MIS 327)

This course is about evenly split between desktop hardware/desktop operating systems and local area networking. The desktop component deals almost exclusively with Intel PCs and Microsoft operating systems. DOS, Windows 3.X & Windows 95 are currently emphasized with some discussion of NT Workstation. Issues such as installation, configuration, upgrades, product selection, and trouble-shooting are covered. The LAN component provides an overview of LAN topologies and protocols, a discussion of LAN hardware such as network cards, cabling types, and hubs. The majority of time during the LAN portion of the course is spent on LAN operating systems. We currently discuss and give demonstrations of peer-to-peer networking using Windows for Workgroups and Windows 95. Most of time is spent on Novell NetWare. An important goal of covering NetWare is to give students some competence in NetWare administration. Two NetWare projects allow students to do this:

Project 1

A NetWare 4.11 server is dedicated to this course. A strength of NetWare 4.X for this sort of course is that its NetWare Directory Services (NDS) allows students to exercise administrator authority within a constricted portion of the network. Although a discussion of NDS is beyond the scope of this paper, it allows for students to create and administer network objects such as print queues, printers, users, groups, organization units, login scripts, etc. for one organizational unit while having no authority to do so for others. Restricted administrator authority for file system objects (e.g. directories) is a feature of almost any networking operating system, including NetWare 4.X.

In project 1, students connect to the MIS server from the general business-computing lab. They use NetWare utilities to configure the server for their "organizational unit". For example, they create user accounts and user groups, write login scripts, set up the file system and assign appropriate rights to users and groups.

Project 2

In this project, students work in teams to set up a small NetWare 4.1 LAN. This LAN includes one server, one client machine and a printer. Tasks included in this project include installing and configuring Ethernet network cards, a CD ROM drive, installing and configuring NetWare on the server and installing either DOS or Windows 95 on the client. The project also includes some administrator tasks that overlap those of project 1. Students are required to get their LAN working so that users can connect from the client, run applications off of the server and be able to print to the network printer.

The combination of these two projects at a minimum, take a lot of the mystery out of LANs. They largely cover the material covered in Novell's 520 NetWare 4.11 Administration course, but also include components from Novell's 804 Course: NetWare 4.11 Installation and Configuration

Workshop. With some additional study, students should be prepared to take the exam for becoming Certified Novell Administrators.

Data Communications (MIS 424)

This course is more of a standard MIS course for which textbooks exist. MIS 327 is a prerequisite. This course is taught at the conceptual level and it includes coverage of LANs, Metropolitan Area Networks and Wide Area Networks. Course topics include the OSI model, network hardware, network topologies, data transmission, standards and protocols, and network management. This networking hardware discussed in the class includes transmission media, modems, multiplexors, Bridges, and routers. Networking standards that are covered include Ethernet, Fast Ethernet, Token Ring, FDDI, TCP/IP, X.25, ISDN, SONET, ATM, and Frame Relay. Students complete a major design project using one or a combination of the technologies. An adjunct instructor who is involved in setting up LANs and WANs for a major regional employer teaches this course.

An Evaluation of the Networking Option

To the extent that students have a sense of what type of job they expect upon graduation when they enter the MIS concentration, networking support is not foreseen as an option. Either through a discovery that programming is not what they want to do, or through interest generated by taking MIS 327 many students decide that network support is something they are interested in. Based on the large enrollments in the elective, MIS 424, many students apparently find the networking alternative to be very attractive. Most MIS students get a MIS internship and many of these positions are in desktop support or local area networking. Through the two networking courses and experience gained on the job, many students are able to find permanent jobs in this field upon graduation. Approximately one quarter of our graduates get into network support in some capacity in their first position when they graduate. On the basis of this, the networking option has been fairly successful.

There are some shortcomings of this program, however. Two courses obviously is not enough to truly qualify someone as a networking professional. It would be desirable, for instance, to be able to cover NT Server in the same or greater depth as Novell NetWare. It would also be useful to have students gain some hands-on experience with additional technologies such as routers and Fast-Ethernet, as well as some Wide Area Networking technologies. Given resource constraints, instructor constraints and the fact that we may not be able to increase the total number of credits required of MIS students, we probably will not be able to alter this much in the short run.

It could be argued that we should focus on NT Server if we were to choose one LAN operating system to cover in depth since many experts believe that NT will become the dominant network operating system. The major problem that was seen in accomplishing this is that the hardware costs were thought to be out of reach. To provide similar projects using NT Server appeared to require several dedicated Pentiums with 32 MB to this class. NT Server does not allow the segmentation of administrator authority on one server to the extent that NetWare 4.X does. A

Novell NetWare Administration course could be taught with just one server. Use of NT Sever, on the other hand, would require that each student have his or her own server to administer. Another factor that makes NT more expensive is that it has more expensive practical hardware requirements than NetWare 4.1 (Pentium with 32 MB vs. 486 with 16 MB). The second NetWare project largely uses 486 machines that have been acquired through upgrading the general-purpose Business lab to Pentiums. The MIS program has been able to acquire the old 486s at no cost. Very little cost has been involved in acquiring the additional needed hardware such as network cards and cabling.

Possible Future Enhancements to the Networking Option

Switch from using Novell NetWare to Windows NT

The major change that is expected is to switch MIS 327 (Microcomputer & LANs) from using DOS, Windows 3.X and Novell NetWare to NT Workstation & Server.. From a marketing standpoint, switching from Novell NetWare to Windows NT would make sense if NT's share of the network operating system market continues to grow.

Although use of NT was thought to be impractical, there may be a medium cost solution to the hardware problem. Another regional college developed a solution to the hardware dedication problem that we may use. They set up hard disk controller that allowed the connection of an external hard drive. Using this approach, all PCs in their general-purpose computing lab were configured to allow the connection of an external hard drive. Each student was assigned a hard-drive for the duration of the course. A student would check his or her hard drive out for class or for doing exercises out of class. A student could use any available PC in the lab and turn it into a dedicated NT server. This solution may be practical for the business computer lab since all PCs will be at least Pentium 100s with 32 MB by the fall 1998 semester. Funds that have been earmarked for acquiring hardware for the MIS program could possibly be spent on this solution.

If the switch were made to NT, there are at least two options for how the curriculum could be structured and delivered. Both options involve offering one or more courses that correspond to the official Microsoft curriculum. These courses are normally offered through Authorized Technical Education Centers (ATECs). Example courses that could be included are Administering Microsoft Windows NT 4.0, or Supporting Microsoft Windows NT 4.0 Core Technologies. The first option would be for UW Parkside to become a Microsoft Authorized Academic Training Program (AATP). This program allows educational institutions to use official Microsoft course materials (class notes, support CD, and lab exercises).. Courses would therefore, adhere closely to the corresponding courses offered through ATECs. The other option would be to teach very similar courses using unofficial course materials. These unofficial course materials include any books on the topic as well as approved Microsoft study guides. With either approach, however, the courses would need to be broadened to include some form of an assessment that can be translated into a course grade.

There are pros and cons of becoming an AATP institution. Doing so would perhaps lend greater credibility to the curriculum and would probably make it more attractive to students. These courses could also benefit from AATP to the extent that the official curriculum provides a good structure and quality support materials. Finally, Microsoft provides free licenses of Windows NT for AATP institutions that offer NT courses. On the downside, some may find it a little offensive that a corporation such as Microsoft dictates a University's curriculum. Similar courses could be offered without the blessing of Microsoft. There is little reason why these courses would necessarily suffer in comparison to those offered by AATP institutions.

Other options

Building on courses taught in the computer science department is one possibility for providing additional learning opportunities for students. Students could be encouraged to take a new web page design course taught in our Computer Science department to enhance their web-related skills. There is a data communications course offered in the computer science department that some students might be able to take. Unfortunately, this course may be of limited interest to MIS students since it focuses on the implementation of networking protocols.

One change that may go into effect next academic year does not involve extending the data communications area per se, but would allow students to gain some experience in developing Web-based applications. We have a course that covers both Visual Basic & PowerBuilder. This course will likely be changed by dropping the PowerBuilder component, and to extend the coverage of Visual Basic to include its use in developing web-based systems. A fairly popular option in industry is the use of VB script in active server pages. This would give students experience developing client/server applications that run on web and database servers using a web browser as the user interface.

Organizing for the University Web Site

Stephen Hawk
School of Business and Technology
University of Wisconsin - Parkside
hawks@uwp.edu

Abstract

Our University has recently begun an effort to improve the management of our web site. This includes both decisions surrounding the organization and staffing of our web efforts as well as reviewing our current web page design guidelines and the extent to which they should become enforced standards. As a result of this, a number of issues have been raised that are probably common to other University web sites. This paper reports on issues faced by UW Parkside in managing its web efforts and reports on recommendations on how to handle these issues.

Organizing for the University Web Site

Stephen Hawk
School of Business and Technology
University of Wisconsin - Parkside
hawks@uwp.edu

Introduction

Our University has recently begun an effort to improve the management of our web site. This includes both decisions surrounding the organization and staffing of our web efforts as well as reviewing our current web page design guidelines and the extent to which they should become enforced standards. As a result of this, a number of issues have been raised that are probably common to other University web sites.

The Current Situation

Our network administrator administers the campus web server, performing duties such as setting up directories, user accounts, assigning rights, maintaining search utilities, maintaining the pages that link to departmental web pages, and putting up minimal department pages for those academic departments that do not develop one themselves. Our public relations director maintains most of the top-level pages that are of interest to students and the broader public. This includes items such as news, announcements, and an overview of the campus and its facilities. School and departmental web page development has been assumed voluntarily by staff or faculty web page developers in the academic and administrative units. There are now only a few academic departments that use the minimal pages developed by our network administrator. There are several holes in the administrative pages in that a number of units have no web page at all.

There have been design standards in place for more than two years. The issues addressed by these standards include colors to be used, image size, use of University logos, providing a return link to the university page, providing the name and email address of the author of a unit's web pages, and identifying the last date the page was updated. These standards, however, are really only suggested guidelines. As a result, many departmental web pages deviate considerably from the standards. Since the public relations director and our network administrator develop the pages at the top two levels of the web site, these do follow the standard.

Issues in Managing Our Web Site

There are issues in managing our web site that are probably similar to those at many other University web sites:

Centralized vs. Decentralized Development:

Some parts of web page development have been centralized in the offices of the public relations director and the network administrator. A majority of web pages on our site, however, have been developed in a decentralized manner by faculty and staff throughout the campus. The quality of these pages varies considerably. A concern is that the design or content of some pages may be an embarrassment to the university, and that many pages are out of date.

The Goals of Achieving a Consistent Look and Feel to Our Web Pages vs. Diversity

Viewed as a marketing tool, it may be desirable for our university to decide what kind of image we should project through its web presence. That is, how should the web site "look and feel", what kinds of messages should be sent about the nature of the campus, what kind of personality should be projected? Departments, on the other hand, may be most concerned about these issues from their own narrow perspective. To convey what a department is like, their personality may need to emerge in a way that differs from the general themes of the University.

Multiple Purposes

The web site has evolved to serve at least three different audiences. These include potential students and employees where marketing the university is the key concern, serving the needs of current students, and providing information to staff and faculty, e.g. meeting announcements, minutes, rules and procedures for governance and administration. This issue is of interest since a single web site may not do the best job of meeting anyone's needs. There is particular concern that we need to do a better job in using the web as a marketing tool. A fourth purpose of the web that is emerging is its use as an instructional medium.

Solutions

The following summarizes the major options that were considered and the likely recommendation.

Staffing Alternatives

None of the following alternatives are seen as something that would prevent departments from doing their own thing if they chose to. Departments could, if they wished, work with a developer to help define the content and design, without doing any of the actual HTML creation.

Hire a Full-time Web Developer

The best long-term solution for handling campus webmaster duties is the creation of a full-time campus webmaster position. This person would work with units and individuals on campus to develop and maintain the campus web site. The person in this position, for instance, could put pages in place for units without them, assist any unit in modifying their pages either by helping them to do-it-themselves or by doing the development. The webmaster should have both technical and design skills and ideally would be able to help with innovative web projects such as the production of web-based courses for distance education purposes. Being able to make the site more "interactive" by incorporating CGI scripts has also been discussed role of the webmaster. The goals of making the site more interactive have yet to be specified. Even if a full-time web developer position is approved, it is likely that the network administrator and the Public Relations Director would continue their current web duties.

Use Current Faculty & Staff as Web Developers

With this option, three current web developers would assume somewhat broader duties. The public relations director would continue to develop the marketing-related pages. A staff member who currently maintains some of the administrative pages would assume responsibility for creating and maintaining pages that target current students. Finally, a faculty member who currently maintains several pages for administration and faculty governance would take on webmaster responsibility for pages oriented towards faculty and staff. What this option accomplishes is to distribute many of the duties of the full time developer discussed in the prior section to these three people. This option is not the preferred option, but is put forth both as an interim solution until a full-time web developer is hired, and as a longer term solution in the event that this new web-developer position is not approved.

Use Part-time Student Workers

A faculty member who teaches a web page development course would identify former students that could be hired as part-time student workers to fill the holes in the administrative and academic pages. They could also assist departments to enhance their current pages. The faculty member would coordinate the students. This option was rejected as a campus-wide solution. Even though it would improve the situation in the short term, it was not seen as a desirable long-term solution. Problems were foreseen in students lacking design skills and in applying different html standards. This latter problem was expected to make these pages difficult to maintain.

Standards Alternatives

The main questions related to standards that were raised include (1) whether the current standards need to be revised, and (2) the degree to which the standards should be enforced. Currently, departmental web developer can put whatever they want on the server. There are no standards that must be met for getting a link from the high-level university pages to their page.

The current standards were left in place, but are intended primarily for the top-level web pages. "Top level" was defined as those pages that reside in the server's root document directory or in subdirectories that come off of this root. Currently, this usually translates into those pages that are within 1 click of the main campus home page. A second, reduced set of standards was recommended for all other pages, except for personal home pages. This second standard merely requires that each page indicate the date of last update and the page's author. To a large extent, these standards will apply to the pages of academic and administrative departments. No standards are intended for personal home pages.

Enforcement of the standards for the higher level pages is not an issue since the two individuals that develop pages at this level currently follow the standards. The second set of standards are expected to be enforced differently depending on whether the offending page is for an administrative or an academic unit. In either case, if an offending page is found, the author will be notified and requested to change the page. If requested changes are not forthcoming in the case of an administrative page, a higher level administrator would be asked get the author to comply with standards. In the case of a page for an academic unit, there will be no enforcement beyond making a request to adhere to standards. The reason for the differential treatment of administrative and academic units is that personalities" of administrative units is less of a factor than in academic units, and the tradition of academic freedom makes the enforcement of standards on the academic side somewhat difficult.

Other issues that came up in the discussion of standards include tracking down dead links and censorship. Although not currently part of the standards, a recommendation is that web pages be kept current. Part of keeping current means that there should be no links to pages that don't exist. A program is being evaluated that, among other things, goes through all documents on the web server and reports on any link that lead nowhere. Information from this could be given to web page authors with the request to remove or update the dead links. Related to censorship, there are currently no restrictions of any kind on the content of web pages. As yet, there have been no complaints about any verbal or visual content of web pages. Where this is expected to come up at some point is in student web pages. All students have 2MB of disk space where they can put up personal web pages. No decision was reached on whether any attempt should be made to restrict web page content. As a fallback position, if any censorship issue arises, a document prepared by the UW system attorney on the general issue of censorship will be referred to.

Alternatives for Serving Multiple Audiences

A proposal exists to create multiple virtual web servers with different domain names. All would use the university's domain name. Letters to identify the different virtual web servers for the different audiences would prefix this. All virtual servers would run on the same machine and would use the same underlying directory structure and the same web pages for the most part. The main difference would be in the top-level home page(s). These would be designed to make it easier for the different audiences to find the information they need. For example:

www.uwp.edu would be designed as a marketing page designed to attract and meet the information needs of campus web page "visitors," such as prospective students and their parents. The top-level page should prominently display two links that would serve to easily guide current students and faculty/staff to one of the following virtual web servers.

students.uwp.edu would be designed to meet the needs of current students. Information would range from the course schedule, library hours, and graduation requirements to upcoming sporting events and what's being served in the cafeteria this week.. The top-level page would display two links to guide students to one of the other two virtual web servers.

facstaff.uwp.edu would meet the information needs of the faculty and staff. This page would have links to business services, personnel information, faculty governance documents, meeting schedules and campus events. It also would include links to the other virtual servers.

Ongoing Oversight of Web-Development Efforts

An ad-hoc group of faculty and staff was assembled to develop the draft recommendations described in this paper. Up to this point, there had been no organized, campus-wide discussion of how web development efforts should proceed. Individuals informally assumed development and decision making responsibility for the conduct of these efforts. While the results have generally been satisfactory, there is a need to have a more organized, representative approach for guiding these efforts. Individuals who play a major role in web site development should not autonomously make major decisions that affect the web site (not all of these individuals agree with this statement, however).. Included as a recommendation is the establishment of a permanent advisory committee. Representatives of appropriate academic, administrative and student groups will be on the committee.

Conclusion

The following recommendations resolve some, but not all of the issues faced by our web site. Having a full-time webmaster will only partially address the problems of decentralized development. Someone could still put up an unattractive or offensive web page. Presumably a full-time developer would reduce these problems through consultation, or by assuming web page development responsibility. Having a full-time webmaster would probably reduce some of the diversity in the look and feel of pages through the same means. The standards, as proposed, do little to address this issue. Diversity however, may not really be that bad in that diversity will not be at the top level and diversity can make things more interesting if done well. Having a full-time web developer may help achieve this. Finally, use of three virtual web servers would seem to be a good solution for meeting the needs of the different audiences.

A Study of Techniques for Introducing Concurrency into List Processing Library Routines in Modern Concurrent Functional Programming Languages

Brent Heeringa
Student
Division of Science and Mathematics
Computer Science Discipline
University of Minnesota, Morris
heerinba@cda.mrs.umn.edu

Scott Lewandowski
Professor of Computer Science
Division of Science and Mathematics
Computer Science Discipline
University of Minnesota, Morris
swl@cda.mrs.umn.edu

Abstract

Erlang is a modern functional programming language with additional features that support explicit concurrency through lightweight processes and message passing. Erlang' s clean semantics and lack of side effects make it possible to develop high quality software that can easily take advantage of the availability of multiple processors. Like most functional programming languages, Erlang provides a number of library routines (e.g. `map`) for processing and manipulating lists. These routines are an excellent target for the introduction of concurrent processing techniques. In this work we explore various methods for introducing concurrency into list processing routines such as `map`. We report on the results of our experiments using the Erlang Development Environment, and discuss which approaches to concurrency are most beneficial given the number of processing nodes available and the properties of the computation (e.g. type of function being applied, size of the input list, etc.).

A Study of Techniques for Introducing Concurrency into List Processing Library Routines in Modern Concurrent Functional Programming Languages

Brent Heeringa
Student
Division of Science and Mathematics
Computer Science Discipline
University of Minnesota, Morris
heerinba@cda.mrs.umn.edu

Scott Lewandowski
Professor of Computer Science
Division of Science and Mathematics
Computer Science Discipline
University of Minnesota, Morris
swl@cda.mrs.umn.edu

Introduction

Functional languages have long been favored for the ease they provide in processing lists of information. Early functional languages, however, were often criticized as being too inefficient. Advances in implementation techniques (such as graph reduction) and hardware platforms have made it possible in recent years to design and implement efficient functional programming languages. Many of these languages (e.g. Concurrent Clean, Haskell, Erlang) provide all of the features of more traditional functional languages (e.g. Lisp) as well as extensions for parallel and distributed processing and/or object-oriented programming [Wilhelm 1995, Plasmeijer 1997]. As efficiency has improved, more and more people have come to recognize the power inherent in modern functional programming languages. Their clean semantics and lack of side effects make it possible to develop high quality software systems that can easily take advantage of distributed processing resources. Modern functional languages are increasingly finding acceptance as a viable software tool for building solutions to real world problems. Erlang, for example, was designed as a language for programming large industrial telecommunications switching systems. It is also suitable for programming embedded real-time control systems [Armstrong et al. 1996].

A key data type in functional languages is the list. These languages typically provide a number of library routines for building, processing, and manipulating lists. Oddly enough, even in modern functional languages that support concurrency, little progress has been made in including concurrent versions of list processing routines (such as map or sort) into language libraries so that the efficiency benefits of concurrency can be easily taken advantage of in building solutions to problems.

Erlang (from Ericsson Telecommunications Systems) is a modern functional programming language that provides built in language support for concurrency and which includes in its libraries a concurrent version of the map function which applies a specified function to all elements in a list. In the next section we describe the map function and our variations in more detail. We then describe our experiments and discuss our experimental results. Finally, we discuss our plans for future work and enumerate the conclusions we have reached. A description of the Erlang language focusing specifically on the features it provides for supporting concurrent computations as well as specific implementation details of our parallel functions can be found in the appendices.

The Map Function

Most functional programming languages provide the ability to apply a function to every item in a list. This function is typically called `map`, and for the purposes of this discussion will be assumed to take two arguments: a function of one argument, and a list. The function is applied in turn to each of the elements of the list and the results of these function applications are collected in a list. Put another way:

$$\texttt{map(f, [}l_1\texttt{, }l_2\texttt{, …, }l_n\texttt{])} \Rightarrow \texttt{[f(}l_1\texttt{), f(}l_2\texttt{), …, f(}l_n\texttt{)]}$$

where `f` is a function of one argument, and where the elements of the list are of the appropriate type for `f`. For example:

```
map(sqr, [1, 3, 5, 7 ]) ⇒ [1, 9, 25, 49].
```

Map can be implemented as a simple linear recursion over the elements of the list `l`. In Erlang, we would express this is as follows:

```
map(F, [H|T]) → [apply(F, [H])|map(F, T)];
map(F, []) → [].
```

Erlang uses pattern matching, a powerful tool for binding values to terms of the same shape. Pattern matching allows a function to be expressed as a collection of case definitions. When defining a function, the order of the case definitions is unimportant due to pattern matching. For example, the base case of map (i.e. the declarative containing the empty list as the second argument) may be positioned last because the pattern for the empty list will in fact be matched when appropriate.

In Erlang there are several patterns which are useful when creating functions that process or manipulate lists:

- `[]` denotes the empty list.
- `[H]` denotes a list containing exactly one term, where `H` is bound to the value of that term
- `[H|T]` denotes a list containing at least one term, where `H` is bound to the value of the first term in the list, and `T` is bound to the rest of the list (note that `T` will always be a list and that `T` may be bound to the empty list).

When used in a context similar to the right hand side of the first case definition above, the Erlang operator `|` indicates concatenation (i.e. the left operand to `|` is concatenated to, or added to the front of, the list denoted by the right hand operator to `|`). Thus `[1|[9, 25]] ⇒ [1, 9, 25]`. The apply routine calls the function denoted by its first argument on the elements of the list it is given as its second argument. Thus `apply(max, [14, 7])` is equivalent to `max(14, 7)`. Note that the length of the list and the arity of the function must be the same.

Tracing through the example above results in the following:

```
   map(sqr, [1, 3, 5, 7])
⇒  [apply(sqr, [1])|map(sqr, [3, 5, 7])]
⇒  [apply(sqr, [1])|[apply(sqr, [3])|map(sqr, [5, 7])]]
⇒  [apply(sqr, [1])|[apply(sqr, [3])|[apply(sqr, [5]|map(sqr, [7])]]]
⇒  [apply(sqr, [1])|[apply(sqr, [3])|[apply(sqr, [5]|
   [apply(sqr, [7])|map(sqr, [])]]]]
⇒  [apply(sqr, [1])|[apply(sqr, [3])|[apply(sqr, [5]|
   [apply(sqr, [7])| [ ] ]]]]
⇒  [sqr(1)|[sqr(3)|[sqr(5)|[sqr(7)|[]]]]]
⇒  [1|[9|[25|[49|[]]]]]
⇒  [1, 9, 25, 49]
```

Since modern functional languages are (typically) side effect free, each application of the function `f` to an element of `l` is independent of all other applications of `f`. Therefore, as long as the results are assembled appropriately, the order of application is irrelevant. This makes`map` a prime target for the introduction of concurrency.

The Erlang library provides a straightforward concurrent implementation of the map function, called`pmap` (i.e. parallel map), where each application of`f` to an element of `l` is computed within its own process (i.e. a separate, self-contained unit of computation). With a sufficiently computationally intensive function and enough processing nodes this approach yields significant performance gains over the sequential implementation. There is however significant overhead associated with this approach – specifically the costs of spawning a potentially large number of processes and of sending a potentially large number of messages over the communications network.

In our experiments we investigate the costs and benefits associated with concurrent implementations of the map function. We examine the performance of the `pmap` routine provided in the Erlang library as well as that of two implementations of our own which we briefly describe below. Both of our implementations borrow a technique commonly used in implementations of quick and merge sorts. When implementing these sorting algorithms, lists aren’ t always broken down to single elements due to the overhead associated with recursion. Rather, the lists are broken down to some pre-determined threshold length, at which point a simpler (non-recursive) sorting algorithm (e.g. insertion sort) is used. We observe that in building concurrent implementations of the list function map it is not necessary to spawn a new process for each list element. Overhead costs can be significantly reduced by breaking the list down to some pre-determined threshold length, at which point the sequential version of map which is discussed above can be used.

Our first approach, which we’ ll refer to as`smap` (i.e. map over sub-lists), is a straightforward modification of the Erlang library routine `pmap`. In this implementation we do not create a new process for each application of `f` to an element of `l`. Instead, we divide the list into segments of length eight or less and create a process for each segment wherein the sequential version of `map` is used to apply `f` to each of the eight (or fewer) list elements. Since fewer processes are created using this approach there is less of an overhead penalty compared to the standard Erlang approach.

In this approach, all of the work involved with splitting the list into segments occurs in oneprocess which we’ ll refer to as the root process. A new child process is then spawned for each list segment of length eight or less. This leads to a process organization similar to that shown in Figure 1. The root process is responsible for reassembling the results from all of the child processes into the final result. For a list of N elements, this approach requires the creation of $\lceil N / 8 \rceil + 1$ processes.

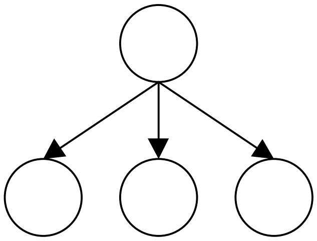


Figure 1: Process organization in `smap`

Our second approach, which we’ ll refer to as`tmap` (i.e. map using a tree split), also splits the list into segments of length eight or less. Like `smap` it creates a process for each segment wherein the sequential version of `map` is used to apply `f` to each of the eight (or fewer) list elements. The key difference when compared with the previous approach involves how the list is split up and where the work occurs.

In this approach we use a binary splitting technique that leads to a hierarchical process organization similar to that shown in Figure 2. As was the case with `smap`, the work performed by the leaf processes is that of applying the sequential version of `map` to lists of length eight or less. Internal processes (shown in gray) are responsible for splitting their input list in half, for spawning two child processes to operate on each half of the list, and for reassembling the results from those child processes into a result. This result is returned back to the node’ s parent process. For a list of*N* elements, this approach requires the creation of no more than $2^{\lceil \log_2(\lceil N/8 \rceil) \rceil + 1} - 1$ processes.

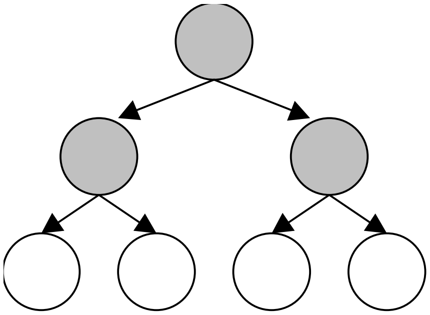


Figure 2: Process organization in `tmap`

Our Experiments

We implemented both approaches described above as Erlang routines and ran a number of experiments mapping functions of various orders of magnitude over lists of varying sizes. Table 1 summarizes the experiments that we report on in the next section.

	O(1)	O(N)	$O(N^2)$	$O(N^3)$
`map`	16, 128, 1024, 16384, 32768	16, 128, 1024, 16384, 32768	16, 128, 1024, 16384, 32768	16, 128, 1024, 16384
`pmap`	16, 128, 1024, 16384	16, 128, 1024, 16384	16, 128, 1024, 16384	16, 128, 1024, 16384
`smap`	16, 128, 1024, 16384, 32768	16, 128, 1024, 16384, 32768	16, 128, 1024, 16384, 32768	16, 128, 1024, 16384
`tmap`	16, 128, 1024, 16384, 32768	16, 128, 1024, 16384, 32768	16, 128, 1024, 16384, 32768	16, 128, 1024, 16384

Table 1: Experiment Summary

To investigate what effect increasing the number of Erlang nodes has on our distributed routines, experiments were run on configurations of two to five nodes. Each node was started on its own workstation – either on one of two Pentium II 300 MHz systems with 128 Mb of RAM or on one of three Pentium 200 MHz systems with 64 Mb of RAM. For each configuration of nodes, mapping routine, mapped function, and list size, we performed a series of twelve runs. In reporting our figures, we eliminated the fastest and slowest of the twelve runs and averaged the results of the remaining ten.

Note that the upper limit on the size of lists used in our experiments is limited in the case of`pmap`. This is due to a limit on the number of processes the free version of the Erlang Development Environment will allow at any given moment.

Our Results

The results of our experiments were, by and large, what we expected, with one small caveat. The graph shown in Figure 3 below is representative of our overall findings. Note that nodes *one* through *three* are the Pentium 200 MHz systems with 64 Mb of RAM.

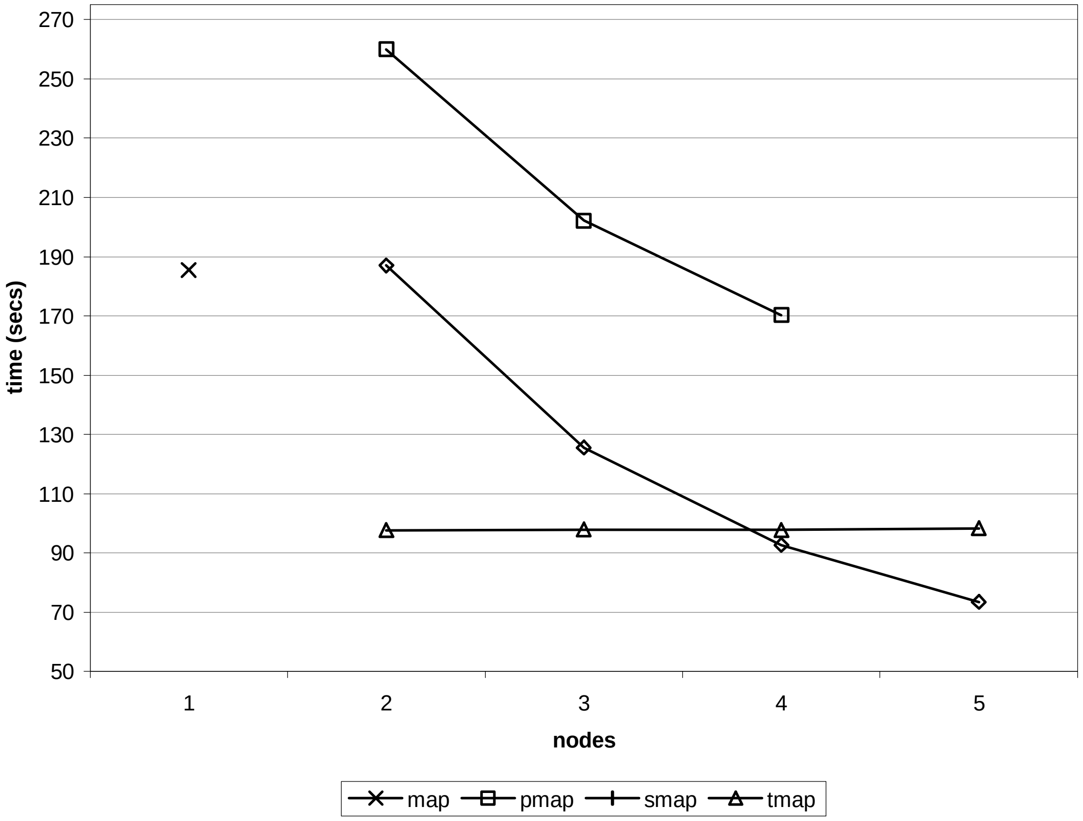


Figure 3: Results of mapping an O(N^3) function onto a 16384 element list

The single cross in Figure 3 indicates the execution time of the sequential version of `map`. As discussed above, the performance of the version of `pmap` provided in the Erlang library suffers due to the significant overhead involved in spawning 16384 processes. The performance of `smap` takes a small initial hit due to overhead when only two nodes are involved, but quickly and dramatically improves as more processing nodes are added. While we are still investigating the reasons for `tmap`' s initial but dramatic performance gains, we do have an explanation for its lack of improvement as more processing nodes are added. The problem lies in how the Erlang library routine `parallel_eval` (which lies at the heart of all the distributed versions of the map function) assigns processes to nodes. Basically, no matter how many processing nodes are available, `tmap` as implemented, will only ever use two of them (see the appendices for more details on the inner workings of our code and the Erlang library routines). The idea that the work is split evenly between two nodes at least partially explains why the performance of tmap in this instance is roughly half that of the sequential version of `map`.

Figure 4 shows the results of mapping an O(N^2) function onto a 32768 element list. The performance trends are consistent with those shown in Figure 3, although `tmap` displays an unusual improvement in its performance with the addition of a fifth node. This cannot be attributed to a different subset of nodes being used, as the experimental results indicate the same nodes would have been used in either case. Since the time difference is only a matter of seconds, a probable explanation for `tmap`' s improved performance, is a decrease in network traffic flow. Note also the graph in Figure 4 does not display the results for `pmap` as they are significantly higher than the times shown, ranging from 68 seconds using five nodes up to 76 seconds using two nodes.

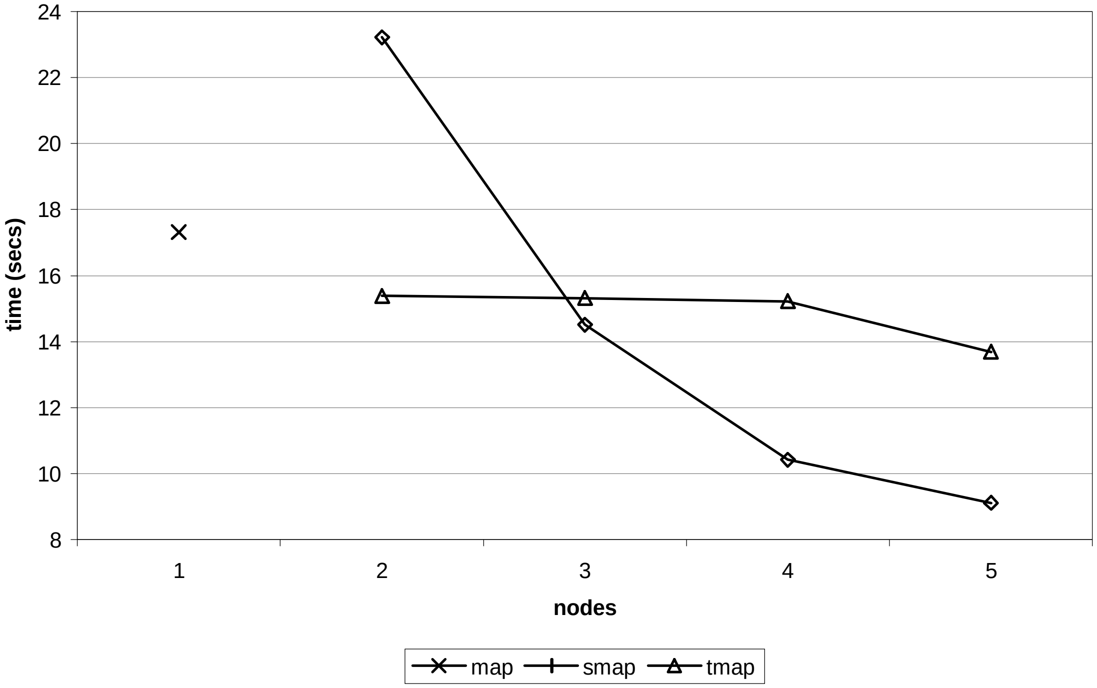


Figure 4: Results of mapping an O(N^2) function onto a 32768 element list

Figure 5 illustrates the consistent behavior of `smap` over all of the experiments outlined in the previous section. Notice the recurring pattern in each set of four columns; as more processing nodes are added, `smap` almost invariably produces results more quickly. Additional experiments, varying the size of the sublists (i.e. using values other than eight), could help determine an optimal relationship between sublist length and performance.

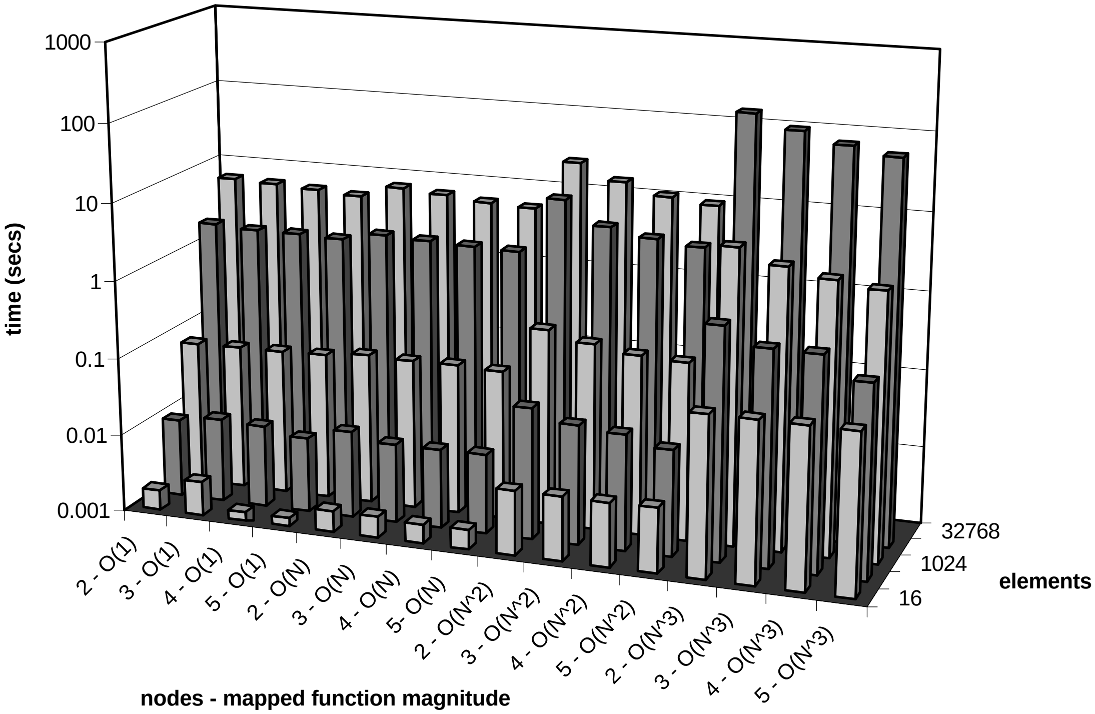


Figure 5: `smap` performance summary

Future Work

One unsurprising result of this research found that parallel computations in Erlang, using the generic `parallel_eval` function, were computationally bound by the slowest processor. This is due to the fact that processes are distributed evenly in a round robin fashion (i.e. job *j* is assigned to node *n mod j*). Given the heterogeneous nature of computing equipment commonly available even within a single lab, we would ideally like to create concurrent routines where faster processors receive a larger portion of the computation. This would replace the even distribution of work currently supported, and allow for even further performance gains.

Erlang provides some simple mechanisms to help with load distribution including the notion of a node pool that is used in conjunction with a load checking algorithm. The Erlang libraries provide a function called `statistic_collector` which performs load checks on nodes in the pool by simply checking the length of the run queue. The run queue is "simply the number of processes which are ready to run" [Armstrong et al. 1996]. Collecting statistics at run time creates an adaptive load distribution which could provide a noticeable speed increase, as well as solve many of the problems associated with the round robin style of process distribution. Notably, the failure of `tmap` to spawn processes beyond the scope of two nodes is solved by adaptive load distribution.

Another interesting tool might be the development of a concurrent evaluator, that is, an algorithm that may determine when concurrency brings about benefits, and conversely when evaluating a function locally is more beneficial. Because of the transparent distribution provided by Erlang, an implementation of such an evaluator function may fit nicely into list processing library routines. For example, the algorithm might check number of nodes available, current run-queue statistics on those nodes, and finally the length of arguments given to the possible concurrent routine as input. The evaluator would be folded into possible concurrent function definitions, as opposed to a standalone function, since its role would not be to determine whether the function itself would benefit from concurrency, but rather if the current computing situation would be beneficial. That is, the evaluator would not determine whether function `foo` would benefit from concurrency, but rather, if the current environment (i.e., number of nodes, current load, etc . .) lends itself well to execution of the function in parallel.

Finally, although this research concentrated on parallelizing one common list processing routine–`map`, it would be trivial to apply such parallelization across many other library functions. For example, common sorting routines such as mergesort and quicksort could reap the benefits of concurrency quite easily if implemented in a distributed manner. Other functions which perform operations over the elements of a large lists, such as `all`–a function which takes a predicate and a list as arguments and determines whether each element in the list satisfies the predicate–may be also be viable candidates [Springer 1989].

Conclusions

The research conducted within the scope of this paper suggests that the addition of concurrent list processing routines in standard modern functional programming libraries would be beneficial. The Erlang shell provides a concurrent-friendly environment where implementation of such parallel routines may take place. Our research demonstrates the ability to easily improve upon current parallel library functions. Our implementation of `smap`, for instance, outperformed the simple `pmap` in all computational experiments. The research also indicates that although using the round robin method of load distribution produces good results, the incorporation of adaptive load distribution techniques would provide even further performance benefits.

Appendix 0: Details of Distributed Erlang as Related to List Processing

Appendix 0 is provided for the reader interested in details related to distributed Erlang and its role in constructing parallel versions of list processing routines.

Erlang provides transparent distribution; that is, the ability to spawn remote procedure calls without general knowledge of other Erlang nodes. An Erlang node is a self-contained unit; an entity that may run standalone or as part of a network of nodes. Not surprisingly, a single machine may run multiple nodes. Alternatively, an Erlang node may be thought of as an instance of the Erlang shell with a unique name identifier. The shell is basically a command line interpreter. Name identifiers are given by the user upon initiation of the Erlang shell and are of the form ‘ nodename@domain.name’ . Nodes with identical magic cookies (a simple, authority driven, security mechanism where a string of alphanumeric characters, called the cookie, is passed along with every remote procedure call) are allowed communication through TCP/IP. Distribution is thus achieved through lightweight asynchronous message passing between nodes.

“Many operating systems provide complex mechanisms such as remote procedure calls, global name servers, etc. as components of their systems. Erlang, however, provides simpler primitives from which such mechanisms can be constructed” [Armstrong et al. 1996]. Implementations of global name and remote procedure servers are included in current releases of the Erlang Development Environment. In this work, we use the supplied remote procedure server in order to specify which nodes receive what processes.

Included in the Erlang libraries are two important modules –`parallel_eval` and `promise`. The first function provides an abstraction for parallel evaluation and the second, a means by which distribution and concurrency can occur without idle wait time.

Erlang’ s`parallel_eval` function takes as an argument a list of 3-tuples. Every 3-tuples consists of a module name, function name, and a list of arguments for the given function. The Erlang syntax for this list of tuples is shown below.

[{mod_name, function, [arg$_1$, arg$_2$, arg$_3$, . . ., arg$_n$]} , . . . ,

```
 {mod_name, function, [arg1, arg2, arg3, . . ., argn]}]
```

The purpose of the argument list is to provide parallel_eval with a pre-determined collection of tasks to be evaluated in parallel. parallel_eval, on the most basic level, assigns these collections of sub-tuples to nodes, executing remote procedure calls on those nodes, using the module/function given in each 3-tuple. parallel_eval also preserves the order of the collections given as arguments. Purposefully, parallel_eval is implemented in such a manner that the programmer decides, through splitting, how modules, functions, and data will be divided into collections.

The definition of parallel_eval is as follows:

```
parallel_eval(ArgL) ->
      Nodes = [node() | nodes()],
      Keys = map_nodes(ArgL, Nodes, Nodes),
      lists:map({promise,yield}, [], Keys).

map_nodes([], _, _) ->
      [];
map_nodes(ArgL, [], Orig) ->
      map_nodes(ArgL, Orig, Orig);
map_nodes([{M, F, A}|Tail], [Node|MoreNodes], Orig) ->
      [promise:call(Node, M, F, A) | map_nodes(Tail, MoreNodes, Orig)].
```

One can see from the function definition that parallel_eval gathers a list of nodes using the nodes()function, appends itself (i.e. through the call to node()) onto the list, and consequently generates a list of promises through map_nodes, using the argument list and available nodes as input.

map_nodes recurses through the argument list, binding the module, function and argument list, of each sub-tuple to variables M, F, and A respectively. It also binds Node to the first term of the node list. It then constructs a list of function calls to promise:call with the previous bound variables using a recursive call with arguments Tail, MoreNodes, and Orig. It is important to note that when the node list becomes empty, it is matched with the second case definition of map_nodes. This declaration then calls map_nodes with the original node list Orig that conveniently has been passed along by the function, creating a round robin style of node distribution. Finally, the yield function is mapped to the list of promises generated by map_nodes (which has been bound to the variable Keys).

A promise solves many problems related to synchronous message passing in that it provides a place holder for returned remote procedure call values, successfully eliminating the need to wait for a process to return in order to continue with evaluation of an algorithm. The Erlang promise module is written as follows:

```
-module(promise).

call(Node, Mod, Fun, Args) ->
      spawn(promise, do_call, [self(), Node, Mod, Fun, Args]).

yield(Key) ->
    receive
        {Key, {promise_reply, R}} ->
            R
    end.

 do_call(ReplyTo,N,M,F,A) ->
      R = rpc:call(N,M,F,A),
      ReplyTo ! {self(), {promise_reply, R}}.
```

The promise module while small, provides important support for efficient distributed computations. The `call` function spawns as a local process `do_call`, providing the caller's process identification number (the call to `self()` returns the PID), the name of the node on which to execute, the module and function to execute, and an argument list as arguments. Note that Erlang's spawn function is analogous to UNIX's "&" in that it returns a PID. `do_call` in turn uses the remote procedure server to evaluate function `F` in module `M` over the argument list `A` on node `N` producing result `R`. It subsequently sends a message to itself (using the "!" operator) which includes the result `R`.

The idea of a promise, or simply, a place holder, results from this explicit message passing. The function `yield` will return `R`, the desired result, only when it has been informed by `do_call` that `R` has been evaluated. Thus, `yield` will only finish evaluating when it receives the proper message, in essence blocking advancement of the function till a message is sent to it. One can see from the above definition of `parallel_eval` that wrapping `yield` around a function call to `promise:call` completes the parallel evaluation process.

Overall, Erlang provides a simple but powerful framework within which it is possible to develop concurrent application routines. The `parallel_eval` and `promise` modules provide good abstractions over concurrency, giving the programmer sufficient control of decisions relating to the distribution of arguments.

Appendix 1: Implementation Details Regarding Various Parallel Map Declarations

Appendix 0 provided significant details of distributed Erlang as it relates to list processing. Appendix 1 will examine various parallel map functions, designed and implemented of as part this research.

As stated previously in Appendix 0, the crux of `parallel_eval` lies within the collection of arguments given to `parallel_eval` in the form of an argument list. For simplicity, we will call the formation of argument lists, splitting. Erlang provides a simple splitting routine for its parallel library map routine (`pmap`) which returns a list containing the normal Module, Function, Argument List triple but restricts the Argument List to a singleton; that is, of the form:

[{mod_name, function, [term$_1$]}, . . . ,{mod_name, function, [term$_n$]}]

As was discussed earlier in the paper, this approach is rather inefficient, creating an unnecessary amount of overhead. Two other splitting techniques though seemed quite plausible, and admittedly, more efficient.

The first `smap`, splits the arguments into sublists of eight terms, thereby mimicking the serial nature of `pmap`, without the overhead of remote procedure calls with single element lists. The split definition for `smap` is declared in the following way:

```
split_args(M,F,As,List,Len,Ack) ->
   case Len < 9 of
     true -> [{lists,map,[{M,F},As,List]}|Ack];
     false -> split_args(M,F,As,lists:nthtail(8,List),Len – 8,
          [{lists, map, [{M, F}, As lists:sublist(List, 8),]} | Ack])
   end.
```

`split_args` takes a module `M`, a function `F`, a list `As`, a list `List`, the length of `List` (`Len`), and an accumulator `Ack`. The `case` statement `Len < 9,` stops the recursion, while any list larger then 9 is split on the first 8 terms, appended to the accumulator, and used as the new `Ack` in the (tail) recursive call to `split_args`. The behavior of `smap` dictates that the non-parallel library routine map, will be executed as a spawned process, as opposed to `pmap`, where the actual applied function is spawned.

The second splitting technique used in `tmap`, performs a simple binary split of the input. The declaration is a bit different though. Instead of giving 3-tuples of the form `{lists,map,[foo]}` or `{mod,applied_func,[foo]}` to `parallel_eval` as input, it sends itself, thereby creating a recursive spawning behavior. As a result, the declaration calling `parallel_eval` must include a base case to stop the recursion. We take advantage of this fact, stopping recursive spawns on lists of length eight or less, then choosing the `smap` style of evaluation through the non-parallel library routine map.

`tmap`, the function which calls `parallel_eval` is defined below:

```
tmap(M, F, As, 0, []) -> [];
tmap(M, F, As, Len, List) when Len < 9 ->
      lists:map({M, F}, As, List);
tmap(M, F, As, Len, List) ->
      lists:append(rpc:parallel_eval(
                   tree_split_args(M, F, As, List, Len))).
```

`tree_split_args` is defined below:

```
tree_split_args(M, F, As, List, Len) ->
      L_O_L = Len div 2,
      [{newtmap, our_treemap,
        [M, F, As, L_O_L, lists:sublist(List, L_O_L)]},
       {newtmap, our_treemap,
        [M, F, As, Len - L_O_L, lists:nthtail(L_O_L, List)]}].
```

Note `div` is Erlang’ s integer division infix operator, and`lists:sublist(List, n)` gives the first n items of list `List`, while `lists:nthtail(n, List)` provides items `n+1` to end of list.

References

[Armstrong et al. 1996] Armstrong, Joe, et al. (1996). Concurrent Programming in Erlang. Englewood Cliffs, NJ: Prentice-Hall Inc.

[Springer 1989] Springer, George; Friedman, Daniel P. (1989) Scheme and The Art of Programming. Cambridge, Massachusetts: The MIT Press.

[Wilhelm 1995] Wilhelm, Reinhard; Maurer, Dieter. (1995). Compiler Design. Edinburgh Gate, Harlow, England: Addison-Wesley Publishing Company Inc.

[Plasmeijer 1997] Plasmeijer, Rinus; van Eekelen, Marko. (1997). Concurrent Clean Language Report. University of Nijmegen.

Acknowledgments

Great appreciation is given to the University of Minnesota, Morris in regards to financial support for this research through the Morris Academic Partnership (MAP) Program.

Many thanks to Jason Anderson for his helpful work in the University of Minnesota, Morris Computer Science lab (a.k.a. The Dungeon).

Also, a quick thanks to the students of Dian Lopez’ s Computer Systems and Organization class for putting up with the times when computers were unavailable due to our research.

Integrating Technology Into Computer Information Systems Classroom Instruction – What Works and What Doesn't

Jay Hettiarachchy

Department of Mathematics and Computer information Systems
Valley City State University
Valley City, ND 58072
hettiara@plains.nodak.edu

Abstract:

This paper describes the experiences of integrating technology into an introductory computer class in a small college. Unlike in the mega-universities, resources available for implementing technology in small colleges are limited. The author's experience teaching similar courses in four different universities for a period over ten years, has enabled him to identify the strengths as well as weaknesses of applying technology in a classroom that deals with a significant aspect of today's technology – computers, learning about them and how to use them productively. The conclusions drawn in this paper provides some useful insights into integrating technology in similar classrooms in other small colleges.

Integrating Technology Into Computer Information Systems Classroom Instruction – What Works and What Doesn’t

Jay Hettiarachchy

Department of Mathematics and Computer information Systems
Valley City State University
Valley City, ND 58072
hettiara@plains.nodak.edu

Introduction:

Integration of technology into a classroom instruction setting could range from planning the physical layout of a classroom, including classroom modeling and lighting arrangements, to enhancing classroom instruction with interactive video conferencing via satellite transmission. Other technological integration, such as video, audio, slide and transparency projection, computer use including local area, Internet, and World Wide Web access could be placed within this range. The experience described in this paper is different from such experience as in a “mega-university” where technology is usually integrated into classroom instruction to cope with large numbers of students, sometimes as many as 1000 in an introductory level Chemistry class without compromising the quality of instruction. Instead, it is the opposite, a small college implementing technology in the classroom to be competitive in providing a quality education in the face of declining student numbers and ever dwindling funds. Unlike in mega-universities, the resources available for integration of technology in a small college classroom are minimal. Nevertheless, the challenges faced by faculty are enormous and many times insurmountable.

Background:

The author’s efforts to introduce technology in the Introduction to Computer Information Systems classroom instruction goes back to 1993 when an institution-wide change in the focus of education was adopted by Valley City State University (VCSU). Started as Total Quality Learning (TQL), the focus was redefined as “Student Centered Learning with a technology emphasis” after a period of one year experimentation with TQL. In the spirit of Total Quality, it is assumed that every student is capable of high achievement.

Being an integral part of technology itself, the Computer & Information Systems discipline provided an ideal experimenting ground for integrating technology in the classroom. However, the path to technology implementation was not altogether free of barriers. There were only a few computer terminals or PC-workstations that had electronic mail capability during this time at VCSU. The absence of a local area network was another significant drawback in providing connectivity among the available computers. The integration of technology in the classroom was limited to the use of a computer screen projector, transparency projector, and three classrooms with stand-alone desktop computers, and a few computer terminals that were connected to an in-house IBM System/36. A few faculty members had access to the North Dakota Higher Education Computer Network (NDHECN) via CMS. Such access was gained by using the terminals connected to the IBM System/36. The available hardware and software, as well as the computer connectivity, were inadequate to support the introduction of significant integration of technology into the classroom during this period. In summary, this period can be called “the period of doing more with less” as far as technology was concerned.

This was also the period when a significant change in the university administration took place. With the retirement of a President and the resignation of a Vice President of Academic Affairs at VCSU, a new university administrative structure that placed two small colleges, Valley City and Mayville State University (MaSU) under a single administration emerged during this period. Similar changes streamlining academic programs with the aid of technology were anticipated to be initiated after.

Although the opportunities for utilizing technology for the enhancement of education by using distance education between these two small colleges situated 80 miles apart were excellent, the available resources for creating the necessary infrastructure to facilitate academic interaction were seriously inadequate. Distance learning was therefore limited to a few classes offered using the Interactive Video Network of the North Dakota University System (NDUS). The absence of a centralized information service between the two colleges was another setback on using distance education methods in the academic programs. Hence the scope of the paper is limited to technology integration into one introductory level CIS classroom within VCSU campus only.

In the Fall of 1996, Valley City State University became a technology intensive “notebook campus.” The University received permission from the North Dakota State Board of Higher Education to increase student fees by $ 475 per semester to finance the acquisition of notebook computers for each student beginning in the Fall of 1996 and to complete campus networking.

The installation of a local area network that enables a large number of students, faculty, and administrators to gain concurrent connectivity locally and to the Internet significantly changed the limitations and placed VCSU in a challenging position by allowing connectivity from 1000 or more locations on campus and over 50 modem connections from off-campus. The “notebook initiative” at VCSU eliminated barriers that prevent many small colleges from maximizing the use of the Internet and the ever-expanding information on the World Wide Web. Making computer resources available to all students at all times was an important step that facilitated technology integration in the classroom.

Technology Integration in the Introduction to Computer Information Systems Class (CIS 180)

Ideally, technology integration should enhance education. Such enhancement should include, and may not be limited to, the following: should enable, a) distance education that allows several sites to participate in an educational experience interactively without losing quality of education, b) teachers to prepare and deliver content material relating to a course in a manner that helps a wide variety of learners with different achievement levels, c) learning material to be available to the students who would like to use such material at all times outside of the classroom, d) learning experience to be assessed for “quality learning.” An attempt will be made in the following section to examine the extent to which the Introduction to Computer Information Systems classroom was integrated with technology along the above lines.

CIS 180, Introduction to Computer Information Systems, is a freshmen level computer class that almost all the disciplines in the university require. The author has taught this class at VCSU and other similar courses in three other colleges and universities in Connecticut, North Dakota, and Minnesota over a period of 10 years. According to the catalogue description of this course at VCSU, the course provides an introduction to word processing, spreadsheet, database, and operating system software. Additional topics include: history, ethics, uses of computers in society, and emerging applications for computers.

In keeping with the current technology trends, a textbook published by Course Technology was adopted by the Business Division that currently coordinates this course. According to Course Technology, “(this textbook) is unique in many ways. It taps into the learning styles of today's students, and empowers students to be self-sufficient computer users." The textbook includes 25 hands-on labs that combine illustrations, animation, digital images, and simulations. These labs guide students step-by-step, present them with quick questions, allow them to explore on their own, test them on their comprehension, and provide printed feedback.[1]

Students met in "hot-wired" classrooms. Every student brought his/her computer to the classroom and had the computer on and connected to the local area network as well as the Internet. Every student could print material to the classroom printer or any other printer anywhere on campus.

Classes were conducted with reduced lecture time, thereby allowing more time for hands-on practice. Detailed description of daily work to be completed were e-mailed to all the students everyday before the class met. Lab assignments were collected at the end of every class period. The lab assignments served as indicators of attendance and consistent work. Two quizzes, a mid-term, a final examination, and a team project were used to evaluate the knowledge on course content and skills learned in the class.

Although several different instructors were involved in teaching this class, and all of them more or less followed the same syllabus, no attempt was made in this study to compare the different knowledge and skill levels of students who attended classes taught by different instructors. The observations reported in this study and the conclusions drawn are based on small-scale classroom research conducted by the author in his classes.

What Works

E-mail helped the instructor to provide an outline of each day's work and each day's lab assignments rapidly. Each day's e-mail saved at least 10 minutes of class time. All the students in the class, whether they were present or absent, knew what was expected to be done in the classroom each class period. Students who missed classes were allowed to submit missed work on the following day. Communication between and among students improved as the semester progressed. One assignment consisted of each student sending e-mail to every student in the class.

The mobility of computers helped students to learn more about the "house-keeping" functions of computers. Most freshmen who had problems with their computers became more comfortable and gained confidence in using their computers as the semester progressed.

Students developed computer skills quickly by following the step-wise labs on word-processing, spreadsheet, and database applications. They also learned the art of navigating the World Wide Web for information and learned how to use Power Point presentation software. The team project involved them with tasks of scanning pictures and importing them into Power Point presentations, and integrating Word, Excel, Access, and Power Point.

In summary, automating certain classroom functions seemed achievable. Since the students received an e-mail from the teacher regarding each days work, most students who had already gained the computer skills at the High School level were virtually unaffected by the presence or absence of the teacher in the classroom. Invariably, such students were able to produce the work output at the end of the class period. Students who had "invested time" with their computers outside classroom were not only able to "trouble shoot" problems, but in some instances, help others who had problems with their computers or getting something done with their computers.

What Doesn't

Although the availability of a computer for each student at all times was a good idea, each students having a computer with which he or she worked while sitting in the classroom was also an undesirable feature that disrupted the class work. The notebook computers in front of each student in the class brought with it new classroom management problems. The following is a somewhat incomplete list of events that take place in a classroom with an average of 30 students in a classroom with computers: a) students take at least a minimum of 5 minutes to setup their computers. b) they start off by checking their e-mail while the class is in progress c) some students have their computer speakers loud d) some listen to their favorite music with their CDs and a set of head phones e) some students have hardware problems and raise their hands asking for help f) some students navigate the WEB g) some students work on a term paper that is not related to the class in progress h) some students do print jobs on the classroom printer i) some students play "solitaire" or similar computer games, totally disregarding what is going on in the classroom. The author created "Ground Rules" in his classes, in order that the classroom disruption be minimized with computer usage during class time. Following are selected from the "*Don't List*" of Ground Rules distributed in the class at the beginning of a semester: 1. Engage in sending and receiving e-mail or broadcast messages during class time (unless the class is on e-mail practice), 2. Interrupt class work with questions regarding your computer problems that are totally unrelated to what is going on in the classroom, 3. Be pretending to take notes with the computer, but be using the computer to navigate the WEB or do pressing work of other classes unrelated to the class you are in, 4. Be sharing hard drives, borrowing and lending others work, 5. Be browsing through other's libraries and helping yourself with their completed assignments etc., 6. The following complaints should be made to the help desk: "My floppy does not work; My screen goes blue and I can't get it to work; I can't remember my password; My computer does not print."

After the introduction of notebook computers in the classroom, the active participation of students in class work dropped remarkably. Students who were mostly busy with their own computers did not pay attention to class work. In a way, notebooks created a barrier between the student and the teacher as well as between students themselves. This situation was to a large extent, due to the efforts that maximized the number of students within a small space. The narrow tables fixed to the floor in "notebook-computer-ready classes" made student movements and interaction within such classes extremely difficult. The resulting seating arrangement of classrooms discouraged team work among students. The interaction and team work that existed between and among the students in computer labs with desktop computers with larger screens that could be viewed by many students at the same time, waned after the adoption of the notebooks with flat liquid crystal displays.

Compared to the classroom with a "talking head" to whom students are expected devote their attention, the "hot-wired classroom" provided a self-paced learning strategy to beginning students, especially in learning certain software skills. Nevertheless, it was clear that students were unable to learn and understand the content and concepts only by practicing step by step instructions they learned from programs aimed at automating knowledge and instruction, as many students were unable and/or incapable of applying the skills they had learned by practicing the labs, to "real world" problems.

Conclusions:

The impact of technology in education has been phenomenal during the past decade. This, in turn has had a ripple effect on education in general and traditional education delivery system in particular. While proponents of face-to-face education have fantasized the virtue of their education method, others have argued in favor of utilizing technology for enhancing education virtually creating a learning environment based on "virtual learning" with no face-to-face interaction between the teacher and the students. The question most educators and education administrators ask today is not whether they want to incorporate technology in their institutions, but how do they want to do it. Most small colleges that are caught in this dilemma are facing a challenge – a challenge to compete in the technology race or to be "run over" by the others who compete successfully.

Technology integration into the classroom requires not only an enormous amount of resources, both material and human, but time and dedication on the part of faculty and students. It also requires a change in the philosophy of education where the control of learning is transferred in the hands of the students who are responsible and mature enough to take full advantage of the power of technology. It creates an atmosphere in which a student could use technology in the same way as a text book anywhere or anytime. However, receiving and sending electronic mail itself will have a limited effect on education, unless the content of the electronic mail has the power of enhancing learning and knowledge of the recipient of the electronic mail. The same is true about the Internet, World Wide Web and the " Virtual University."

It was also obvious that learning with technology had the most impact on students who were motivated to learn. Such students needed the least amount of supervision or "policing" by the instructor. Attempting to deliver instructions using the traditional lecture method in a hot-wired computer lab was not only ineffective, but inefficient. Self-learning with programmed instructions interspersed with traditional methods of instruction focused on monitoring the degree of learning, proved to be a better method of enhancing the learning process as well as validating the outcome of the learning process. In the final analysis, however, the net worth of any method of education, whether traditional or non-traditional, technology-based or not, could only be determined by criteria that illustrate the achievement of expected results without degradation of quality.

References:

1. [Parsons and others 1996] Parsons, J. J., & others (1996). New Perspectives on Computer Concepts. Course Technology, Cambridge, MA.

Virtual Lecture, Virtual Laboratory or Virtual Lesson?

Curt Hill
Mathematics Department
Valley City State University
Valley City, ND 58072
curt_hill@mail.vcsu.nodak.edu

Brian M. Slator
Computer Science Department
North Dakota State University
Fargo, ND 58102
slator@badlands.nodak.edu

Abstract

The need for computer based education and distance learning systems is becoming increasingly obvious. In addition, the value of "active" versus "passive" learning has become increasingly clear. Two complementary implementations of this approach are described -- the virtual lecture approach and virtual laboratory approach -- as well as discussion of future plans for integrating these approaches into the Virtual Lesson.

The Virtual Lecture approach implements an Exploratorium-style museum metaphor to create a hyper-course in computer programming principles aimed at structuring the curriculum as a tour through a virtual museum. The Virtual Laboratory approach implements a synthetic planet where Geology students are engaged in a mineralogically based exploration.

Current plans for these research efforts include the integration of the Virtual Lecture with the Virtual Laboratory into the synthesis of the Virtual Lesson which combines the best elements of each.

The Need for Computer Assistance in Education.

Education, like many other disciplines, needs computer-aided assistance. Most educational endeavors focus on an instructor, in a classroom, on a campus, at a particular time. Administrators prefer large lectures because that minimizes cost in an era of increasingly tight budgets. The ones who lose out in this scheme are the traditional students, who get very little instructor contact, and those non-traditional learners who cannot take the time to travel the distance to those campuses where knowledge is dispensed.

Education is very labor-intensive. Numerous organizations have applied technology in one form or another to make the instructor more efficient and the process more cost-effective. Diversity University (www.du.org) is a well known example. This paper suggests another approach.

Toward the virtual lecture

Consider the most common means of teaching in this country: the lecture. In one session the teacher may give out information in a cost-effective way. Unfortunately, this method has some serious deficiencies. Lectures are very much teacher-centered, the teacher is elevated as the only one who really has something to share and is proclaimed the master of the students' destiny. Communication is typically uni-directional, the instructor speaks and the students listen. If well done, in a small group setting, it can be quite good with student involvement, interaction among all members of the group, a sense of ownership. Yet, the practical realities are that a small group is an expensive luxury. Instead, the lecture model views students as passive information sponges. Students are to be quiet, hanging on every word the teacher utters and carefully copying every single mark recorded on the blackboard. There is to be no comprehension at this time, for if a student considers an item for too long the instructor is on to a new topic and they have missed forever some valuable information. On the contrary, their task is to faithfully record every bit of information and, while outside of the classroom, attempt to make sense of it all. In rare cases a student might ask a question during a lecture, but self-consciousness prevents most of these. The problem, even with small group settings, is the experience is more passive than active, and the value of "active" versus "passive" learning has become increasingly clear (Reid, 1994).

There are many students whose learning style makes it quite difficult for them in a lecture setting. They have been branded as slow learners, in high school and shuffled into vocational programs that are much more hands on. It such a situation many of these flourish, not because they were incapable of the higher reasoning needed, but because coincidentally the vocational programs happened to match their learning style. It is no wonder that many people decry the use of the lecture mechanism, yet what is there to replace it?

The content of a lecture can be communicated by a number of means, to make it even more cost effective. Instead of concentrating the students in a single classroom, use some broadcast media, such as television to disseminate the image and sound to a much wider audience. This still has the disadvantages of a single time and the expense of broadcasting, but can effectively communicate nearly as well as a live performance. A videotape can capture the performance, but may be harder to distribute than a video signal. It does have the notable advantage that the student now has control of the time of presentation. Some of the power has been wrested from the instructor and given to the student. However, the presentation itself is very much linear. The student must start at the beginning and process it sequentially. The fast forward, rewind and pause buttons give them some options not available in a real lecture, but the format is still essentially linear. The corresponding disadvantage is the experience is almost totally passive.

The Hypertext Reference Document

The ubiquitous Internet can make a contribution in active learning. The lecture can be reduced to text and graphics and then put in a succession of world wide web pages. Although, they can still be arrayed in a linear order, hypertext may be structured in many other shapes as well. A link to an example may be viewed or skipped. Now the learner has control of many aspects of the presentation. They choose which pages to visit and in what order; the experience is now primarily in the hands of the learner. In this respect, a collection of web pages may be superior to either a traditional text book or a traditional lecture. To be sure, there is a loss of interest as we proceed from the teacher speaking live in a classroom, to the captured image on tape, to merely text and graphics on a web page. The web page is not as engaging as the lecture, but the web page is learner-centered: the learner has control of the process. However, browsing through hypertext is still largely passive in nature.

One possible implementation is, as has been suggested, a collection of web pages. Each page contains a small amount of text and supporting graphics. Each page is then linked to appropriate other pages. The links then structure how a student can browse; this approach is similar, in certain respects to the way traditional museums are organized. In the museum metaphor, each page is an exhibit.

There are some obvious advantages to this type of implementation that need not be belabored here. HTTP servers are quite common, with most higher education institutions already having a server. Authoring web pages is relatively easy. Many applications, such as word processors, have some facility in this area. The hard part is not the web authoring, but that the content is technically and educationally appropriate. The web is available continually to anyone, anywhere; distance education is becoming reality.

The Virtual Lecture

Despite the obvious advantages of the world wide web, it is still a relatively passive mechanism. The active learning alternative is the synthetic environment where learners can experience their education in a "learn by doing" way. The Virtual Lecture approach implements an Exploratorium-style museum metaphor to create a hyper-course aimed at structuring the curriculum as a tour through a virtual museum. Student visitors to the Virtual Lecture museum are invited to participate in a self-paced exploration of the exhibit space where they are introduced to the concepts of the particular domain, are given demonstrations of these concepts in action, and are encouraged to manipulate the interactive exhibits as a way of experiencing the principles being taught.

Virtual Lectures are implemented using freely available MOO environments (for example, the Xerox PARC LambdaMOO (Curtis, 1992) or the High Wired Encore MOO (Haynes and Holmevik,1997)). A MOO is an Object Oriented MUD where a MUD is a Multi User Domain. The most common use of these has been for role-playing games and the like. The metaphor of a MOO is of a text based virtual reality, with rooms and players. In this kind of educational setting a MOO room corresponds to a web page. When a student enters the room, an amount of text is presented in a way analogous to the browsing of a web page. The exits to the room are marked and the student may weave through the museum in the same way as the web. This minimal use of the MOO is equivalent to the hypertext reference document.

However, exploring a MOO is quite different from browsing a series of web pages. MOOs can be much more interactive. A MOO allows the interaction between one student and other students, interaction between the student and the environment, and between a student and software agents. Furthermore, record keeping is possible and even easy in a MOO in contrast with a web implementation. Using a MOO, the capabilities of a web reference document can be enhanced into a virtual lecture.

A MOO contains all the capabilities of an Internet chat room. Thus, students are aware of each other's presence in the room. They may inquire as to whom is presently in the museum and where these people may be. If either other students or instructors are present in the room then they may enter into a conversation. This conversation may be public, received by everyone currently in the room or private, only heard by two participants. A person not in the room may be paged, that is signaled that a conversation is desired and where the originating person is. A student (or a teacher) may disguise their identity. This has the effect of freeing them to ask questions they may not be brave enough to make if their identity is known. Two students at different physical locations may thus communicate and engage in collaborative work, which would be impossible on a web site. The student has all the control they had in the web implementation and substantially more as well. A MOO may also contain software agents. These are programs that occupy the MOO in approximately the same way as a student. They may interact with the student in the form of a tutor or peer.

Record keeping in a web site is very difficult. A server may record what page was served but the identity of whom received the page is usually only an IP address. In contrast a MOO requires a real login and password. Because of this login process, a variety of record keeping is kept automatically and more is available with special programming. It is therefore possible to track students' progress through the museum.

A Virtual Lecture Implemented

A specific example of a Virtual Lecture is the Programming Land MOO at Valley City State University which is being developed as an adjunct to programming classes. The MOO contains material that parallels an introduction to programming in C++. The MOO server is called WinMOO (Unkel, 1997) which is a port of the original LambdaMOO server from UNIX to Windows NT. The original database was the enCore database, which is the LambdaMOO database enhanced with numerous objects of educational merit, such as moderated classrooms, lectures, etc. The server is version 1.8.0p6 and database is enCore Beta 2.

Student visitors to the Virtual Lecture museum are invited to participate in a self-paced exploration of the exhibit space where they are introduced to the concepts of computer programming, are given demonstrations of these concepts in action, and are encouraged to manipulate the interactive exhibits as a way of experiencing the principles being taught.

The museum is being used with one class in the fall 1997 semester and will be used with two in the spring. There are two MOO entryways to the Virtual Lecture. The first leads to a series of rooms that describe the basic commands of the MOO. The other leads to the C++ Foyer exhibit. This takes the student to one of several topics. Each of these lessons may lead to one or several further lessons. A lesson is an amount of instruction that could reasonably be completed in one sitting, whereas a topic is usually several lessons and hence too large for a single session. It should be noted that both lesson and topic are arbitrary terms without specific boundaries in the MOO. If a student wants to learn one lesson in several settings, they have the freedom to progress at their own pace in whatever way they choose. Thus, students do not perceive lesson boundaries or topic boundaries. All they see are single exhibits, which are single rooms and the brief amount of information that is present in that context.

A single exhibit will convey a very limited amount of text. This text may be any of several types. One common exhibit is a signpost. A signpost exhibit does not convey much technical information, instead it is usually the entrance to several other lessons and topics. The C++ Foyer is a signpost directing students in any of several directions. Figure 1 shows the Function Lesson which is a signpost exhibit and an example of what a touring student would actually see in this Virtual Lecture:

```
Function Lesson
This is the start of a number of lessons about functions.
Consider the following menu that may be selected by letter or
topic:
 a) The importance and usefulness of functions (why)
 b) Using functions or calling functions (call)
 c) Overview of function definition (define)
 d) Function parameters (parms)
 e) Function return values (type)
 f) Function and variable scope (scope)
You may choose any of these and enjoy.
Obvious exits: [exit] to C++ foyer, [define] to Function
Definition, [call] to Calling Functions, [parms] to Function
Parameters, [scope] to Scope Lesson, [why] to Why use
functions?, [type] to Types that functions produce
```

Figure 1: an example of a Signpost Room

The first line “Function Lesson” is the MOO room name. The next ten lines are the room description. The last four lines are a listing of the exits from this room. This description as a signpost does not convey any significant technical information, but does direct students to a series of lessons. This particular signpost is arrayed like a menu. A student may type either “b” or “call” and go to the same room. The name of the room is given when the room is created. The description of the room is written after the room is created and is the main means of conveying information. The last four lines are generated by the MOO server to indicate the available exits. Each of these exits may have one or more synonyms that cause the student to progress to the destination room.

The most common type of exhibit is informational, a room where some content is given. This can take any of the forms that a lecturer would use. For example on the menu of Figure 1 the first option is mostly motivational -- why is this feature useful to the student. In an actual lecture this is needed to pique the curiosity or otherwise show the need of the concept about to be discussed. Lecturers do not need to motivate the good students, but a lecture has to be inclusive, so motivational comments are required for a good lecture. Yet in the virtual lecture students may pick and choose what they view and in particular a student may skip the first part of the function lesson if they desire. Other rooms may have the informational content of other parts of a lecture: simple descriptions, a variety of longer descriptions and examples. No oral lecture can have the number of examples of a virtual lecture since the student does not need to view them all, only enough to grasp the concept.

The example of the Virtual Lecture so far presented could be easily implemented with a series of web pages: a hypertext reference document. The next part considers some of the aspects of the MOO that are not readily implementable with web browsers and these enhance the mostly passive hypertext reference document into the active Virtual Lecture.

The MOO attaches to each student a list of exhibits they have visited. This list of exhibits is available to the instructor and is a record of progress and a diagnostic tool for that student. This list cannot consider comprehension, but does demonstrate exposure. For distance learners it is a concrete measure of activity for a student that may have no other communication with the instructor. A student who has a problem, may just have missed the part of the Virtual Lecture that dealt with the item in question, so that an obvious course of remediation can then be suggested.

This history of exhibits also enables an improvement to the structure of the museum. In order to keep the hypertext reference document convenient many more paths must be created than a novice should be allowed to use. Designing a Virtual Lecture requires a balance between the single linear path, which penalizes advanced or returning students, and the potential for too many choices which invites the novice to exhibits where they are more likely to be confused than educated. The solution to this dilemma is the active exit.

The active exit checks the prerequisites for the room. A student taking a path to an advanced topic has their history checked against some possible background exhibits. If the student has visited rooms that form the foundation for the room to be entered, they are allowed in without knowing that their prerequisites have been tested. If they have not visited the rooms needed then the exit suggests that this room may be more confusing than helpful and asks them if they really want to proceed. An advanced student may have acquired the needed knowledge outside the Virtual Lecture and thus can go forward, while a merely curious novice has been warned and at least knows there is potential confusion ahead.

A very important consideration for any educational tool is the interest level that the tool maintains in the student. Consider public libraries: most of those that lend video tapes are actually lending more tapes than books, which is especially remarkable when considering the selection of book dwarfs that of video tapes in most libraries. The implication is clear, video tapes are more engaging than books. Similarly, the Exploratorium model for a museum is more interesting than the older museums with glassed in exhibits and no interaction. If interest level was not a consideration we would write good textbooks, hand them to the students and administer examinations at the end of the time. The impracticality of such an approach is obvious. A web implementation of the reference document will hold a student for a while, but will eventually become essentially the reading of an electronic text. What is needed is something even more interactive than the reference document or even the Virtual Lecture possible in a MOO.

In the web implementation this may be accomplished by Java applets, but in the MOO this will be accomplished by software agents. Such an agent may appear as a code machine that occupies the room and has various commands that operate it. It may also be a robot, that the student might or might not distinguish from another student, who comes alongside and questions or tutors the student as they walk through the museum.

The Programming Land MOO at present has but one agent, a code machine. The code machine is an interactive object placed in appropriate exhibits. The main job of the code machine is to demonstrate short pieces of programming language code. It accepts several commands:

1. **Show** lists the lines of code
2. **Explain** gives an explanation of each line in succession
3. **Trace** shows the execution of each line in succession
4. **Next** advances an explanation or trace by one line
5. **Help** explains the possible commands of the machine.

This object is intended to raise the interaction level and engage the interest of the student in a an active way impossible for a textbook or web page.

This Virtual Lecture does not eliminate the utility of a web site for the class, both complement each other. The class web pages contain a general page about the entire course which includes the announcement of tests, links to assignment documents, the source of programs discussed in class, answers to exercises, and overview documents like the syllabus and course outline. The normal web browser makes it easier to download documents than most MOO clients. Therefore, the web is a very handy administration tool for the course, but the Virtual Lecture is far better for instruction.

The Virtual Laboratory

Simulated environments are valuable teaching tools because they can take a learner to places they would never ordinarily experience — either because they are too dangerous, or because they are a physical impossibility. GAMES are Graphically Advanced Multi-player Educational Simulations that provide authentic environments for Reality Oriented Learning Experiences (ROLEs). The players in a ROLE-based environment actively participate in a sustained problem-solving simulation. To succeed in these virtual worlds, and to effectively play GAMES, a learner will necessarily master the concepts and skills required to "play their part". Role-based learning is learning-by-doing, but not the mere goal oriented "doing" of a task. Role-based learning is learning-by-doing within the structure and context of playing a role. Rather than simply teaching goal-based behavior and task-oriented skills, the ROLE-based approach communicates a way of practice (Slator and Chaput, 1996)

The Geology Explorer (Slator et al, 1997), is an interactive multi-media educational game for teaching Geoscience in a role-based, goal-oriented, and learn-by-doing way. The purpose of the project is to study several issues connected with highly graphical and highly interactive learning technologies with a view towards developing effective teaching systems and efficient methods of implementation. To accomplish these goals, design and development have gone forward on four fronts: 1) the assessment of student uses of educational games; 2) the development of a theoretical construct for explaining student use (not discussed here); 3) the prototype development of an experimental game; and 4) the design of a suite of software tools for continuing the development cycle (also not discussed here).

Although computer technology has caught the imagination of many educators, it has yet to be conclusively proven if the technology increases student learning of course material. A report entitled “Is Distance Learning any Good?” (TeleEducation NB, 1995) provides one reference list of publications that found technology had a significant impact on learning and a second list with references that found that technology had no significant impact. To provide more definitive evidence, though, courseware tools that are rich in content and have high interactivity must first be built before any assumption regarding the effectiveness of computer tools to improve learning can be tested.

As part of the Geology Explorer development, experiments are being conducted so as to answer particular research questions concerning 1) the effectiveness of these educational environments in terms of student learning and technology use, and 2) the effectiveness of our approach to game design and development.

The over-arching goal is to discover ways to design and develop educational software that is both learning- and cost-effective. In the pursuit of this, we address several issues in educational game design, student learning, and educational technology uses.

The research questions fall under two headings. In particular our experiments address the following educational research questions:

1) "What do students learn from these systems?"
2) "How does the role-based approach contribute to student learning?"
3) "What style of learning do these systems best support?"
4) "How do these systems compare to more traditional methods?"

and the following design and development questions:

1) "What are the effective procedures for designing and developing role-based, game-like educational software?"
2) "What are the software tools needed to expedite the design and development of subsequent systems?"

From a development point of view, this research project is composed of the following elements:

1) Virtual Laboratory development to simulate the objects of Geoscience and Planet Oit;
2) Graphical user interface design and development to customize the Virtual Laboratory client software for this application;
3) Development of the assessment protocol; and

4) Development of a role-based theory of enculturation in order to support the development of future designs and tools for implementing future environments.

Developing methods for the assessment of student learning is a central element of the research. The subject group is students in a freshman level Geology course. Physical Geology (NDSU Geology 120) is an large-enrollment (>400 students/section), 3 semester hour lecture course. Aside from lecture, the course content is augmented by slides, by a set of course lecture templates, by a textbook, and by a web resource site which includes self-quizzes, photographs, course news, and links to related resources. Testing is by multiple choice exams, with students submitting their results on optical scan sheets. Nearly 100% of the students enroll in the course to complete either general education requirements or specific course requirements within their majors. Of the 434 students enrolled at the beginning of Fall, 1996, none were Geology majors.

The challenge for the instructor is to try to make these non-science oriented students think like a scientist: proposing and testing hypothesis, making appropriate decisions utilizing the basic tools of a geologist, working with the language of geology as a science, etc. It is obviously impractical for an instructor to take 434 students into the field and have them individually experience how a geologist makes on-site decisions. However, the student can experience these in a Virtual Laboratory, in which each student would act as a geologist and be expected to address a series of plausible geologic situations. Within this environment, the student would make decisions similar to those of a "real" geologist using the tools and techniques of a geologist.

In addition, because it is tied to an Earth-like planet, the course has a geographic base upon which to establish this synthetic environment. The base is not limited to just the Earth's surface but to the relationship of this surface to processes and energy sources deep within the planet. Synthetic environments could then be established at any position on, above or within the Earth sphere, and the student can evaluate diverse and often counteracting forces and processes.

The students of Physical Geology 120, or a portion of them, are to be subjects in a bi-modal study. Each will be given a pre-test instrument composed of two parts:

1) a selection of standardized, multiple choice questions; and
2) a subjective, case-based exam where authentic geoscience scenarios are described and the students are asked to work through the problem sets as a geologist would.

The students will be separated into two groups at some point in the semester. Half will be given a self-paced, on-line, textbook and quiz system to work through. The other half will become explorers on Planet Oit. At the end of the parallel sessions (we imagine this will be a 2-week mini-course undertaken in the Fall of 1998), a similar two-part post-test instrument will be administered. With this data, we will undertake to answer the educational research questions listed above.

The Geology Explorer (Slator et al., 1997) is a synthetic environment to simulate a portion of Planet Oit (very similar to Earth, and in the same orbit, but directly opposite the Sun). Students "land" to undertake an exploration exercise armed with tools and instruments implemented as Virtual Laboratory objects. They are given an authentic geosciences goal, e.g.. to locate and report the position of potentially valuable mineral deposits. Accomplishing these goals will entail mastering several geoscience concepts and procedures, and will demonstrate student mastery of the material. The first module involves mineral exploration, where students are expected to plan an expedition, locate mineral deposits, and survive the somewhat hostile environment in order to report on it.

Planet Oit is comprised of over 50 locations from which the geological expedition can begin. Geological tools and instruments have been developed (such as streak plates, hammers, and geiger counters), and the appearance and response of 40 minerals and 40 rocks to a series of interactions are described.

Students are transported to the planet surface and are issued or otherwise acquire a standard set of field instruments (a rock pick, a bottle of acid, a magnet, and so forth). Students are issued an electronic log book to record their findings and, most importantly, they are assigned an exploratory goal. These goals are intended to motivate the students to view their surroundings with a critical eye, as a geologist would. In the context of the Geology Explorer, these goals are chosen from a principled set, in order to leverage the role-based elements of the game.

The students make their field observations, conduct small experiments, take note of the environment, and generally act like geologists as they work towards their goal of, say, locating a Kimberlite deposit, or assessing soil mineral content for arability. A scoring system has been developed, so that students can compete with each other and with themselves.

Planet Oit is in the process of being populated with intelligent tutoring agents to monitor, mentor, and remediate the human learners in the performance of their roles. These agents are developed as sub-topic experts who have access to problem solving experiences, context sensitive help and advice, conceptual and procedural tutorials, and stories of success and failure within their particular sub-topic. The agents monitor player's progress and "visit" a player when they need their particular help. The agents coach the players by sharing their expertise in the form of prototypical case studies, problem-solving dialogs, and pre-packaged tutorials.

Tutoring agent behavior is adaptive in two senses. First, each agent is the owner of a small set of sub-topics and related cases. As a learner operates in the synthetic environment they will be building their own case, and the relevant agent will be alerted for remediation whenever a learner case becomes similar and relevant to a case under the agent's control. As learner behavior changes, so will their profile and the nature of the cases they match against. In this way, agents will gain and lose interest in a player according to the changes in the learner's profile.

Second, learner state will be preserved throughout the course of their involvement of the synthetic environment. As learners leave the game, either as successful or unsuccessful players, their state and experience will be saved as a new case. These saved cases, according to their profile, will become part of the inventory assigned to one or more of the tutorial agents. As later players enter the synthetic environment, the tutorial agents will have these additional cases of success and failure to present as part of their remediation package. In other words, tutorial agents will begin the game armed with prototypical case studies, but they will accumulate additional student case studies as players enter and leave the game over time.

In addition to these plans, Planet Oit has an existing inventory of "deductive problem solving" agents which monitor student actions and give advice, but they do not mandate or insist on student actions, nor do they block or prevent student actions. There are currently two types of tutoring agent in the game, and plans for a third (for more complete details, see Slator et al., 1997).

The equipment tutor detects when a student has failed to acquire equipment required to achieve their goals The equipment tutor is called when instruments and tools are acquired. The tutor checks whether the student has the instruments they need to identify their rock or mineral goal. If not, the tutor visits with the player on that topic. Future plans may also entail the tutoring decision to remediate when buying instruments that serve no obvious purpose.

The exploration tutor detects when a student has failed to observe a goal in the course of their travels. The tutor checks whether the student is leaving a room that might contain a rock or mineral goal. If so, the tutor visits the player to inform them. Future plans may entail a REDUCTION in tutoring behavior, if it should be found that too much tutoring intervention is an obstacle to student learning. In other words, the tutors may wait for several failures to occur before remediating.

The science tutor is the next to be implemented. It will detect when a student makes a deductive mistake in identifying their rock or mineral goals. There are four cases to be considered:

(wrong tests) the player has "guessed" incorrectly and their history indicates they have not conducted the necessary experiments to identify their goal

(wrong answer) the player has "guessed" incorrectly and their history indicates they have conducted the necessary experiments to identify their goal (lucky guess) the player has "guessed" correctly but their history indicates they have not conducted the necessary experiments to identify their goal

(good work) the player has "guessed" correctly and their history indicates they have conducted the necessary experiments to identify their goal

The Science Tutor has knowledge of the learners goals, and knowledge of the "experiments" needed to confirm or deny the identity of a goal. For example, suppose the student is given the goal of locating and identifying talc used in the production of cosmetics and other materials. To confirm that a mineral deposit is indeed talc the student must test the deposit with the "streak plate" and observe a white streak and scratch the deposit to determine its hardness is less than 2.0 on the standard scale.

The system will encode the necessary and sufficient experiments for each goal, as well as their expected results. The system will check these facts against the student's history to determine when to remediate.

The Virtual Lesson

Synthetic environments, such as those described above, are dynamic and extensible spaces. These environments support multiple users (so learners can interact with both the environment and each other), real-time simulation of events, and interactive software agents, particularly tutors. This flexibility permits complementary approaches to student tracking and modeling, as well as mentor-style interactions where an "over the shoulder" software tutor monitors student actions and "visits" participants with timely and felicitous help and advice.

The eventual goal of these efforts is to combine the Virtual Lecture and the Virtual Laboratory and produce the Virtual Lesson. A Virtual Lesson will be an active environment where things can happen quickly. As soon as a student learns a concept in the Virtual Lecture, they can apply it in the Virtual Laboratory. Indeed the boundary between the two will dissolve.

Related Work

Far and away the most common approach to implementing synthetic multi-user environments is the text-based MUD: the multi-user, text-based, networked computing environments that are mostly for "gaming". MUDs, or Multi-User Dungeons, are an outgrowth of computer chatlines and bulletin boards plus the popularity of adventure role-playing as exemplified by Dungeons and Dragons. They are environments which one can log into from a terminal connected to Internet, and then interact in text with objects, places, and other players within a gamelike setting (Carlstrom 1992).

The Social Virtual Reality project at Xerox PARC has extended MUD technology for use in non-recreational settings. Their goal is to keep the strength of MUDs --- shared computing with a powerful real-world metaphor --- while correcting their shortcomings by adding audio, video, and interactive windows. They have built two specific prototypes: Astro-VR, for use by the professional astronomy community, and Jupiter, for use by researchers within Xerox. (Curtis and Nichols, 1993)

In a recent search of the World Wide Web it was clear that MOOs for different ability levels are becoming a reality. Amy Bruckman, a doctoral student at the Massuchessetts Institute of Technology has built a programming language to make it simpler for children to construct objects and participate in MOOs (Bruckman, 1993). She has combined construction and community in the hope of creating a constructionist learning culture in her MOOse-Crossing MOO.

The Donut MOO in Stark County, Ohio addresses the needs of K-12 students. Students build the MOO by creating "textually anchored virtual reality spaces." Although the site is open to all students, many are older.(Suzie 1995)

MOOs have shown their importance in elementary schools. Two in particular, MariMuse, and MicroMUSE have been geared so that elementary school students can participate full-time. One noteable success has been on underachieving students who had left school. These students reportedly became involved, started to form friendships, and began to take a greater interest in school.(Poirer 1995)

In some learning environments, the virtual reality MOO connected to a network is already here. At the United States army base and training facility in the Mojave desert, there is presently a virtual reality multiuser computer simulation that can be linked to other military bases all over the world. Using topographical resources and mapping facilities, the entire Mojave desert has been recreated digitally. Soldiers from all over the world can participate in the same wargame scenario.

Other examples of virtual reality MOOs, sometimes called "multi-user computer simulations,", are being implemented in a virtual physics classroom being developed at NASA, and with interactive programs run the Loma Linda Medical centre in southern California.(Mclellan 1994). Mclellan (1994) cites some early conclusions about the VR MOO experience with other entertainment games such as "Battletech".

Mineral Venture is a recently developed software environment that simulates business-oriented mineral exploration from a technical and economic perspective. This is not a multi-user spatially oriented exploration system, but rather a simulation intended to pose planning and resource management problems that geologists routinely face.

SELL! is a multi-playered, networked game that teaches basic marketing and micro-economic concepts. Players are immersed in a simulated environment where they are expected to save a failing retail outlet. The tools of the retail trade, (hiring, advertising, ordering, pricing), are made available, and the underlying simulation is crafted to respond to game play in plausible ways. Throughout the course of a game, players have the opportunity to consult real world entrepreneurs, advertising executives and economists for guidance as they attempt to build up the net worth and market presence of their simulated businesses (Slator and Chaput, 1996; Hooker and Slator, 1996).

The Geology Explorer project (Slator et al, 1997) implements an educational game for teaching the Geosciences. This takes the form of a synthetic, virtual environment, Planet Oit, where students are given the means and the equipment to explore a planet as a Geologist would

Acknowledgements

The following people were involved with the Geology Explorer as a class project in its earliest stages: Scott Gordon Aagard, Elson Abraham, Brian Raymond Allrich, Tonia Marie Brezina, Michele Lea Chown, Murali Dhandapani, Michael Glen Dunkle, Brent Alan Ellingson, Chad Justin Elliott, Bryan James Fugere, Syed Habibullah, Jeffrey Allen Haugen, Robert Jerome Hoffman, Jim Sam John, Beau James Douglas Kautzman, Shane Nelson Kullman, Jason T Lagge, Debra Ann Meyers, Rupa Mitra., Prashanth Mylvarabatla, Curtis Wayne Ophoven, Xiaofeng Pan, Mingbo Qin, Sylvia Salas, Darin Jeffrey Schmitz, Samuel E Silverthorn, David Chadwick Teigland, Daniel Douglas Ward, Yue Yun.

The Geology Explorer team includes Donald Schwert*, Bernhardt Saini-Eidukat*, Jon Abel, John Bauer, Brian Gietzen, Nathan Green, Tammy Kavli, Lucas Koehntop, Bhaskar Marthi, Vidyalatha Nagareddy, Acey Olson, Yongxin "George" Jia, , Kishore Peravali, Daniel Turany, Brad Vender, James Walsh.

Additional thanks is due to the members of the Educational Media Seminar for discussion and feedback: Radha Balakrishnan, Murali Dhandapani, Uma Kedla, Satyanarayana Pasupuleti, Sisir Ray, and Hong Wu.

The Geology Explorer project was supported in part by a Grant from the NDSU PPRG to the NDSU WWWIC adminsitered by Dr. Paul Juell and Dr. Phil McClean, and by NDSU Grant-in-Aid #1109.

References

Curtis, Pavel (1992). Mudding: Social Phenomena in Text-Based Virtual Realities. Proceedings of the conference on Directions and Implications of Advanced Computing (sponsored by Computer Professionals for Social Responsibility)

Haynes, Cynthia , and Jan Rune Holmevik, eds (1997), High Wired: On the Design, Use and Theory of Educational MOOs. University of Michigan.

Reid , T Alex (1994) Perspectives on computers in education: the promise, the pain, the prospect Active Learning. 1(1), Dec. CTI Support Service. Oxford, UK

Slator, Brian M. and Harold "Cliff" Chaput (1996). Learning by Learning Roles: a virtual role-playing environment for tutoring. Proceedings of the Third International Conference on Intelligent Tutoring Systems (ITS'96). Montréal: Springer-Verlag, June 12-14, pp. 668-676. (Lecture Notes in Computer Science, edited by C. Frasson, G. Gauthier, A. Lesgold).

Slator, Brian M., D. Schwert, B. Saini-Eidukat, P. McClean, J. Abel, J. Bauer, B. Gietzen, N. Green, T. Kavli, L. Koehntop, B. Marthi, V. Nagareddy, A. Olson, Y. Jia, K. Peravali, D. Turany, B. Vender, J. Walsh (1998). Planet Oit: a Virtual Environment and Educational Role-playing Game to Teach the Geosciences, Proceedings of SCCS98. Fargo-Moorhead, April.

TeleEducation NB (1995) Is Distance Learning any Good? http://tenb.mta.ca/anygood/

Unkel, Christopher (1997). WinMOO. http://www-personal.engin.umich.edu/~cunkel

Advanced Microcontroller Interfacing Laboratory

Bryan Infanger
Department of Electrical Engineering
North Dakota State University
infanger@plains.nodak.edu

Dr. Robert Gammill
Department of Computer Science
North Dakota State University
gammill@plains.nodak.edu

Prof. Val Tareski
Department of Electrical Engineering
North Dakota State University
tareski@plains.nodak.edu

Abstract

This paper describes how to set up a laboratory to provide convenient access to microcontroller hardware for doing exercises in hardware interfacing. Besides the traditional access given to the student via lab stations, hardware and software has been added to our set up to make much of the laboratory available on the Internet. With expanded campus networking being the rule rather than the exception, it makes sense to allow student access to laboratories by way of the network. Also, the increasing numbers of "older than average" students and students with disabilities lend further impetus for doing this.

Microcontroller hardware available to support a laboratory setting typically consists of an evaluation board connected via a serial port to a PC. Laboratory assignments typically address hardware issues such as parallel and serial ports, interrupts, and timers. Some assignments deal with controlling attached devices, such as stepper motors, LED displays, and DC motors. Described in this paper are the developments necessary to make many of these features available remotely over a network.

The advent of inexpensive personal computers running a free UNIX and X-window has made possible a setting where a student accesses a microcontroller in one window, an assembler and editor in another window, a simulator with unique debugging facilities in a third window, and possibly an editor for composing the laboratory write-up in a fourth window. With network-wired residence halls students could very easily set up such a workstation in their own rooms.

In our CS/EE course on assembly language and computer architecture, several assignments use our microcontroller laboratory that is equipped with Motorola 68332EVK boards. Two major assignments are a round-robin process scheduler that uses real-time interrupts and a stepper motor control exercise. In our EE course on microcomputer interfacing, students use a laboratory equipped with Motorola 68HC11EVB boards. Exercises in this course focus on hardware interfacing issues. For example, one assignment involves running a 4-digit LED display using two output ports. One port contains the data to be displayed while the other port controls latches that select which LED displays that data.

Using an appropriate interface it is possible to put a microcontroller laboratory on the Internet using real processors (not simulators). Students are able to complete many assignments remotely without ever setting foot in a lab. This microcontroller programming laboratory "on the network" has received favorable reviews by the students. When given the choice, some students still go to the laboratory to get a "hands-on" feel of the equipment while others are content working from home. Regardless of how the student chooses to do the exercises, the quality of access to the laboratory has been improved at minimal cost to us.

Advanced Microcontroller Interfacing Laboratory

Bryan Infanger
Department of Electrical Engineering
North Dakota State University
infanger@plains.nodak.edu

Dr. Robert Gammill
Department of Computer Science
North Dakota State University
gammill@plains.nodak.edu

Prof. Val Tareski
Department of Electrical Engineering
North Dakota State University
tareski@plains.nodak.edu

Introduction

In the future almost all electronic products will include microprocessors in the design. Utilization of microprocessors / microcontrollers in a hardware design augments the design's capabilities while simplifying the design's implementation. Due to this it is important that students of Electrical Engineering and Computer Science have undergraduate experience with microcontrollers. It is sometimes difficult however, for a university program to provide an adequate microcontroller lab due to limited lab space and budget restrictions.

In an effort to overcome these difficulties a laboratory setup has been developed utilizing existing networks, campus hosts, and low cost components that allow students superior access to lab equipment, at a price that is reasonable. The enhanced configuration provides this improved access without placing any additional restrictions on the types of laboratory exercises that can be performed.

Microprocessors vs. Microcontrollers

The defining characteristics differentiating microprocessors and microcontrollers can be hazy. At the core, microprocessors and microcontrollers are very similar. The primary operation of both is to process a series of instructions that make up programs, and as such, they both do this in much the same way. Their typical difference lies in that microprocessors are oriented towards higher MIPS, MFLOPS, etc., while microcontrollers are designed with interface circuitry embedded into the chip that provides things like I/O ports, A/D converters, and RS-232. Note that this is the *typical* distinction. Chip manufacturers sometimes blur this distinction when making bigger and better chips.

At any rate, this paper is about an advanced *microcontroller* lab, so the descriptions herein will focus on microcontrollers. However since microprocessors and microcontrollers are so similar, the setup is applicable to microprocessors as well. In fact, on the NDSU campus microcontrollers are used to teach microprocessors.

Background

Before detailing the setup of our advanced laboratory it is helpful to consider some of the issues related to teaching a microcontroller course and what sort of lab options are typically available to instructors.

Microcontroller Issues

There are two aspects to teaching a microcontroller course: the programming side and the interfacing side. The programming aspect is where teaching microcontrollers is similar to microprocessors. To insure student understanding of microcontroller operation, microcontroller programming must be taught at the machine language level. Because of this, additional instruction in concepts such as memory fetches, cycles per instruction, stacks, internal registers, and program counters is also required. These items are avoided in most introductory programming courses and thus make teaching microcontroller programming more difficult.

The other aspect to microcontrollers is the interfacing side, which itself has two facets. First, there is the physical / electrical connection between the microcontroller chip and external device being interfaced. This requires attention to issues of common grounds, voltage/current limitations, noise shielding, and firm physical connections. Even if a student has covered these concepts in other courses, because of limited prior experience, most students taking a microcontroller course need to be reminded of them.

Once the hardwired connection is made, the communications aspect of the interfacing needs to be addressed. For this, students must learn about memory mapped I/O (memory addresses whose value represents and/or manipulates signals pins on the chip), timing considerations (many students put in-sufficient delay in their programs so that the program talks to the device faster than it can listen), and interrupts. Because of the number and complexity of these concepts, a student requires lab experience to ensure understanding.

Lab Types

There are a number of different course counterparts that constitute a lab, albeit some deviate greatly from the classic laboratory image. The first style of lab is to offer computers with microcontroller simulation software. Students can compose assembly language programs and then execute them in a simulated environment. Typically the simulator restricts students to programming that doesn't involve interfacing, with coding assignments requiring only memory manipulation and simple text manipulation operations via simulated monitor routines. More sophisticated simulators offer simulated interfacing where students can observe or stimulate simulated ports. This style of lab isn't great, but it at least allows students a place to write and test their code. It is advantageous because the test equipment is software which, could potentially be copied to all available computers, providing a large number of lab stations for students. Access is further improved if the software is available for a network accessible campus host. If a simulator is placed on such a host, not only would it be available from the campus computers but also potentially from a student's home if the campus offers dial-up lines.

Some of the drawbacks of this lab style are that it doesn't offer hands on experience and doesn't expose students to the interfacing issues. For instance, students running code on sophisticated simulators may assume the code is correct because the simulated output is properly sequenced. However, students might overlook timing issues; even though the output is properly sequenced, it is driving the I/O ports way too fast, and were the code attempted with an actual microcontroller, the communication would fail because the controller is talking faster than the device could listen. If students had developed the code on an actual microcontroller this oversight would have been caught rather quickly and at the same time, taught a valuable lesson. Another drawback to a simulator-based lab is that simulator development often lags chip development significantly. A microcontroller course using a simulator-based lab would be restricted to teaching only microcontrollers for which a simulator was available. This is a problem if a school is attempting to teach the most current technology. Finally, simulators may contain programming bugs, which could cause the simulator to deliver non-authentic responses and hamper a student's understanding.

A second style of lab setup involves using evaluation boards. Evaluation boards are printed circuit boards that contain an actual microcontroller and a minimal set of memory chips and other supporting hardware necessary to make the microcontroller function. Evaluation boards are offered by chip manufacturers to promote their chips and provide customers an easy way to evaluate a new product. Despite their commercial intent, evaluation boards are very useful in education.

Evaluation boards typically run a monitor program that communicates via a serial port. To use an evaluation board in a laboratory a computer is required to communicate with this serial port. From the computer the user can talk to the microcontroller and issue commands to upload code, execute programs, and read and write memory locations. To perform an experiment using this setup, students would enter their microcontroller program into the computer, assemble it, upload the result from the assembly to the evaluation board, and issue a command to begin execution. One advantage of this setup is that the code is being run on an actual microcontroller so that the responses are immune from programming bugs encountered with the simulator. Also, evaluation boards typically come with other connectors that directly route to the various I/O ports, A/D ports, etc. that are featured on the microprocessor. From these connectors students can very easily attach various devices and perform interfacing experiments. Thus as a lab setup, the use of evaluation boards remedies all the shortcomings of the simulator-based laboratory.

Despite being an improvement, the use of evaluation boards does have some disadvantages, the foremost being cost -- evaluation boards are typically more expensive than a microcontroller simulation package. As such, it is typically infeasible to have one lab station per student as was possible with the simulator-based lab. In addition, evaluation boards typically require one dedicated computer for each board. Though the simulators require computers, the computers used can be located in a general purpose, general access computer cluster. Since evaluation boards are expensive, it is usually desired that the computers used to attach to their serial port be located where access is restricted to only students taking the microcontroller course. Because of these additional costs, an evaluation board-based microcontroller lab typically supports only one lab station for every eight to ten students, resulting in contention amongst students for access to the equipment.

A third laboratory setup is actually a hybrid of the other two and stems from a recognition that all experiments need not be done on actual hardware, especially for early assembly language programming experiments that have nothing to do with interfacing and where speed is not an issue. This laboratory orientation involves using a simulator for the initial part of a course and then later using evaluation boards for the actual interfacing and real time experiments. This setup maximizes the number of lab setups at the start of the semester as stations are only limited by the number of general purpose computers available (which is potentially unlimited if the simulator is available on a campus host). This setup can also help alleviate contention for the evaluation boards later on in the semester as students can be encouraged to iron out the bugs in their code using the simulator, and only after those bugs are ironed out, attempt work on the actual hardware.

One seeming drawback of this hybrid solution is that it is not a cost-effective way of doing things since it involves obtaining both a simulator and a set of evaluation boards. This isn't always the case though. One of the expensive parts about using an all evaluation board setup is the cost of the dedicated computers necessary to support restricted access. In an educational environment, there are often multiple courses being offered using specialized equipment needing computers with restricted access. If this is the case, two courses can be orchestrated to use the same lab without interfering with each other. So rather than just a microcontroller lab there is instead, say, a microcontroller and PLD laboratory, helping to justify the computer's cost and the dedicated lab space necessary to house them.

The Advanced Microcontroller Laboratory

The Advanced Microcontroller Laboratory results from years of observing what works best. It is a setup that is the current step in what has been a series of small improvements to an existing laboratory arrangement. It takes the best parts of the *typical* microcontroller laboratory setups and puts them together in such a way that offers very convenient access to equipment without restricting the types of exercises that can be assigned. The design has additional merit in that it is both scalable and modular, and it can be tailored to the needs and resources of any related course.

The Setup

What is needed to set up an Advanced Microcontroller Laboratory is a networkable terminal, a network, a host that allows multiple logins (a PC running a free UNIX would do), a terminal server, and an evaluation board with an appropriate assembler that can be ported to the host. These are the minimal requirements to setup an Advanced Microcontroller Laboratory. The listed items don't need to be purchased or obtained anew as existing equipment will work. Most institutions already have a networkable terminal (typically a PC with telnet capabilities), a campus network and a host, leaving only the terminal server, evaluation board and assembler, all of which can be obtained for less than $500.

The first step in putting together this lab is to obtain an assembler for the microcontroller or microprocessor being featured in the course. The assembler needs to be set up on the host. If the assembler isn't written for the host it needs to be ported, or another assembler found. This shouldn't be a problem as most popular microcontrollers / microprocessors have many assemblers available; some vendors even supply the source code. In the worst case, an arrangement might need to be made with the producers of the assembler (companies are often helpful towards colleges and universities).

Once the assembler is placed on the host that is accessible via the network, students have the ability to compile their programs from where ever they can log in. If working with the minimal requirements for the Advanced Microcontroller Laboratory this would be the *networkable terminal*. However more often there is a networked computer cluster that students use to check their email, or alternatively a set of dialup modem lines, anyone of which would allow students to log into the host and assemble their programs.

The next setup item for the Advanced Microcontroller Laboratory is the terminal server. A terminal server is a network device that attaches to an ethernet, is assigned an IP, and has a number of serial port connections for attaching serial communication devices. These serial ports might be used to attach to some external modems. Incoming modem connections would be able to operate via the IP number assigned to the terminal server, in essence creating a modem bank. However for the proposed lab setup, the evaluation board's serial connection is attached to one of the terminal server's serial ports, and that port of the terminal server is configured as an output port (in the modem example it would be configured as an input port). With this in place, students can telnet from the host to the evaluation board (via the terminal server), or in fact, students can telnet to the evaluation board from anywhere (it need not be from the host) see [Fig.1].

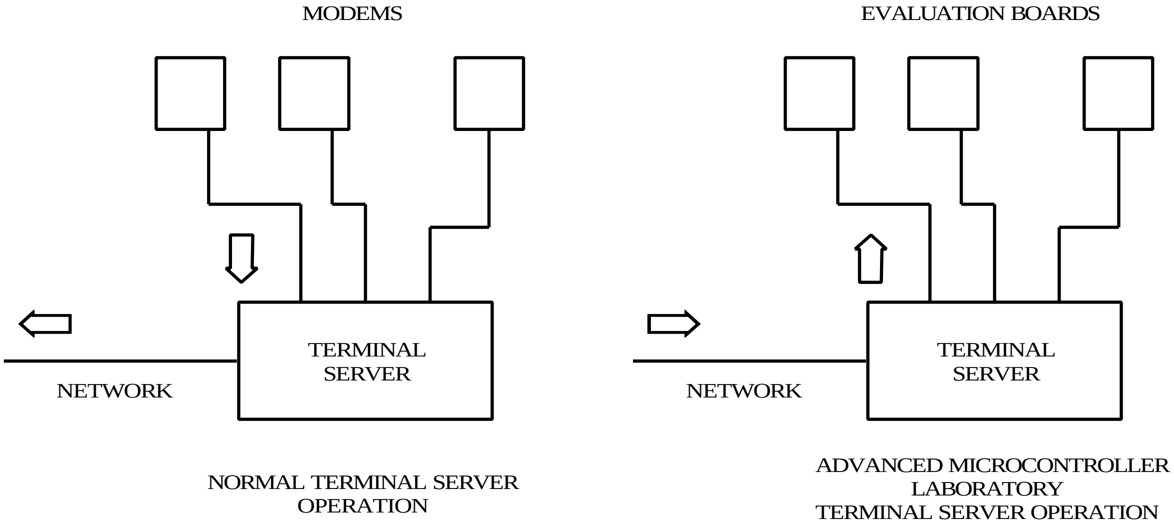

MODEMS
EVALUATION BOARDS
TERMINAL SERVER
NETWORK
TERMINAL SERVER
NETWORK
NORMAL TERMINAL SERVER OPERATION
ADVANCED MICROCONTROLLER LABORATORY TERMINAL SERVER OPERATION

Figure 1: Terminal Server Operation

Once the above is configured you have an Advanced Microcontroller Laboratory [Fig 2]. The merits of this design will become evident in the next section which describes how it is used.

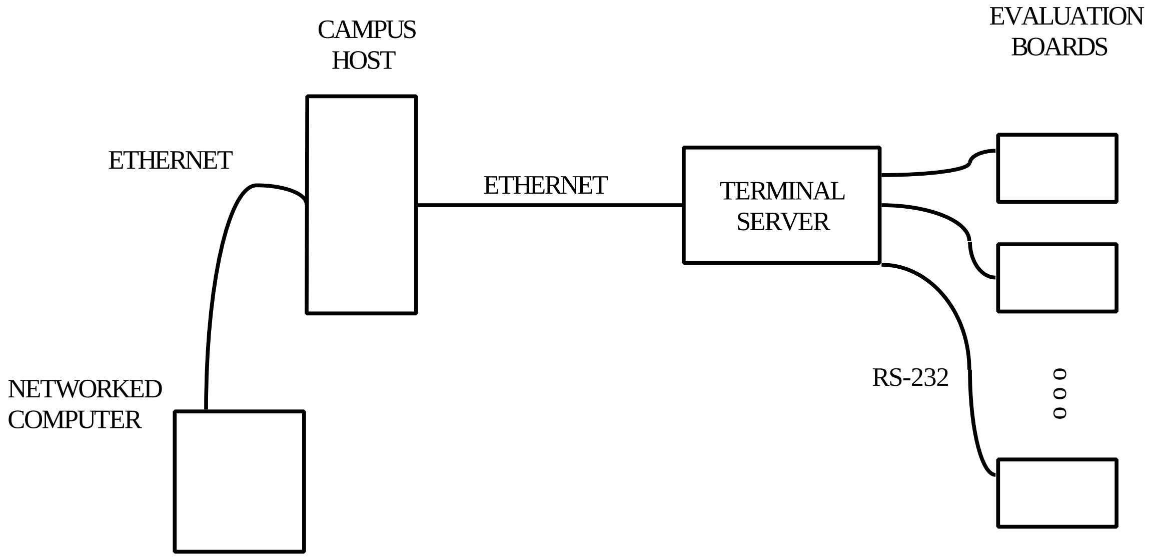


Figure 2: Block diagram of Advanced Microcontroller Laboratory

The Use

The above description details the minimal configuration necessary for an Advanced Microcontroller Laboratory. However, a more typical setup would be to have several evaluation boards rather than just one, and at least a computer cluster or two, plus dial-up lines, rather than just one networked terminal. For purpose of the following description it is assumed that this is the case.

For the first description of how to use the laboratory setup, consider the example where a student has been assigned a problem that involves writing a microcontroller assembly language program that *doesn't* involve interfacing. In this case, the student would first telnet to the campus host from a computer, be it from home or in a cluster. The student would compose a program using one of the host's online editors and when completed, assemble the program. Once the program is assembled, the student then telnets from the host to an evaluation board via the terminal server. The student issues commands to the evaluation board and host to upload the assembled code. Once the code is uploaded, the student executes the program and sees if it delivers the desired results. If the program doesn't deliver the desired results, he or she can debug as necessary.

Some of the pluses of this arrangement should already be apparent. First, evaluation boards are being used, which are actual microcontrollers and thus are free of potential programming errors. Second, most of the time spent working on program is in the composing and debugging stage. The previous example showed that the editing being done used only the connection between the student's terminal and the host, leaving the evaluation boards open for use by other students. Were this a classical setup using evaluation boards, each evaluation board would be attached to a computer, students working on their program would tie up the computer connected to the evaluation board (and consequently the evaluation board), even if they were merely typing their program in.

For the second example the situation is altered such that students have to complete an assignment that actually involves interfacing. In this case some additional assumptions are made about the arrangement of the evaluation boards. It is assumed that the evaluation boards attached to the terminal server are in a lab station style arrangement in close proximity to some computers or networked terminals. In this case the students go to one of the computers or terminals and telnet from there to the campus host. They then take their assembled code and upload it to the nearby evaluation board that is hosting their interfacing experiment. Once all the connections to the microcontroller are ready, students issue commands to the evaluation board to start execution of the code. It is observed whether or not the experiment works. If so, great; if not, then minor adjustments can be made through the computer's connection to the host.

As with the last example, one advantage the Advanced Microcontroller Laboratory has over classical arrangements is that students don't need to tie up the evaluation boards during their initial program composition. With this arrangement, students can do the bulk of their coding from home or from one of the computer clusters and only when they think they have a working product, go into the lab to test it.

Another advantage of the Advanced Microcontroller Laboratory is the fact that all of the students' files remain on the campus host under their own account. In conventional setups where evaluation boards are attached directly to a computer, student files will collect on the computer's hard drive, accumulating as junk and providing students a convenient means to use another student's code.

Details

In the previous section a number of details were skipped for the benefit of understanding the gist of the operation. This section details those and other items.

An important issue in the use of this laboratory is determining which evaluation board students end up being connected to. In the case of the non-interfacing example, it doesn't really matter, as students merely need a microcontroller to test their code. In the interfacing case, however, students will have presumably set up some additional hardware to a specific evaluation board, so it is very important they get connected to that board as they will be watching the interfaced devices for reactions.

This issue is handled by the capabilities of the terminal server. As mentioned before, the terminal server has two sets of connections: one is the ethernet connection that allows it to communicate with the host, and the other is the collection of serial ports that allow it to connect to the evaluation boards. Associated with each serial port is a number that maps to an IP port that can be telneted to. The serial port number plus an offset determine the IP port number. For example, if there was a lab station, numbered station one, that possessed an evaluation board that was connected to serial port number one, a student could connect to that specific evaluation board by issuing a command from the campus host like, "telnet terminal.server.edu 2001". A virtual connection would then be established and the student could then upload their assembled program and run their interfacing experiment.

The terminal server also provides a feature called a *rotary* that facilitates connections when it doesn't matter which unit is connected to. Without this feature a student wanting to connect to *any* evaluation board would have to manually attempt to connect to each known unit, a potentially frustrating activity if many of the boards were in use. With the rotary that is not a problem; students merely attempt a telnet connection to a special port (the port number of the rotary) and the terminal server will automatically connect students to the first available evaluation board. If none are available, the terminal server instead delivers a connection refused error message, something that would still be frustrating, but not nearly as much as they just attempted to connect to a dozen evaluation boards only to find out that all of them were being used.

The second item that was skimmed over was how programs are uploaded to the evaluation board once a connection is established. As mentioned, the evaluation boards are intended for a serial port connection, and as such, it is expected that programs will be uploaded via that serial channel. When assemblers assemble a program the output generated is typically in a special ASCII format. For the Motorola microcontrollers we are using this is the "S-Record" format. The S-Record format was designed for uploading programs to microprocessors on evaluation boards. S-Records encode both the opcode data and the associated addresses in an ASCII printable format.

A program is uploaded by issuing a command to the evaluation board's monitor telling it to expect S-Records on the serial line. The monitor then waits indefinitely for S-Records to arrive and will not stop listening for S-Records until it sees a special S-Record that denotes the end of transmission. Once the monitor is placed in this wait state, the user then issues commands to the telnet client to transmit the contents of a specified file (which would be the resulting file from the assembler). Once the entire file has been transmitted, the monitor returns to a command prompt. The user is then free to issue a command to the evaluation board to start program execution.

The final item that needs to be mentioned is a specially-designed break detect circuit. It is possible for a microcontroller program to contain an infinite loop; a set of instructions that will cycle over and over until the processor is reset or until otherwise interrupted. This is not a problem if a student is working in lab, where they can just hit the reset button on the evaluation board. However with the Advanced Microcontroller Laboratory it is likely that the student won't be in lab, but rather will be working remotely, perhaps at home. If the evaluation board gets caught in an infinite loop it will become unusable. Since it would be inconvenient to have to travel to the lab to reset the microcontroller every time it locks up in an infinite loop, a means to reset the evaluation board remotely is required.

The solution developed is to attach a tiny unit inline with the serial connection that is wired directly to the evaluation board's reset line. The inline unit is designed such that when it sees a break signal on the serial line it pulls the reset line of the evaluation board low, causing the unit to reset. The inline unit ignores any other signal or information, responding only to the break signal.

Types of Laboratory Experiments

There are a myriad of experiments that can be performed with the advanced microcontroller laboratory. Some of our favorites follow.

Round Robin Scheduler

Using a Motorola 68332EVK style evaluation board in our CS/EE course on assembly language and computer architecture, one of the assignments is to write code to implement a round robin process scheduler. This assignment utilizes a Real Time Interrupt (RTI) generator (that is embedded in the microcontroller) that sends periodic interrupt signals to the main processor at intervals determined by the contents of a specific hardware address.

The objective of this assignment is for students to write code that mimics an operating system's job scheduling duties by divvying CPU time between two different processes running concurrently. To do this students must write code that will suspend processing of one task and resume processing by way of storing and retrieving register images. The round robin scheduler switches processes every time it receives an interrupt from the RTI generator.

To provide validation that the scheduler is doing what it is supposed to be, the final phase of the assignment requires students to develop a print routine that uses the evaluation board's monitor routines for producing output. These routines are accessed by way of TRAP instructions. A TRAP instruction provides a convenient way for an assembly language program to make a system call.

Driving a Stepper Motor

Another favorite assignment in the CS/EE course on assembly language and computer architecture is to write code to drive a stepper motor setup. Stepper motors are special motors with multiple coils that represent phases. The phases need to be strobed in a specific order to induce motion in the motor. The ordering is typically to turn one phase on while turning off the adjacent phase. Each transition between phases results in a *step* of the motor. Stepper motors are often times used as actuators, where small precise motor movements are required.

In the stepper motor setup used in this experiment, wires associated with each phase of the stepper motor are attached to one of the 68332's output ports. Since the 68332 uses memory mapped I/O, this allows a program running on the 68332 to cause movement in the motor by adjusting the value stored in a specific memory location. With this, the first part of this assignment is for students to get the stepper motor to turn in both a clockwise and counter clockwise direction.

The stepper motor setups also incorporate a photodetector mechanism that is wired to an interrupt generator on the 68332s. This attachment is setup such that when the stepper motor reaches a certain point in its rotation a signal is sent back to the 68332 causing an interrupt. The second part of this assignment requires the students to modify their code such that the stepper motor turns in one direction until an interrupt occurs and then reverses direction, resulting in an oscillation. This oscillation continues until the motor completes some predetermined number of steps.

One of the challenges associated with this experiment is discovering the sequence of numbers (corresponding to phase strobes) necessary to cause movement in the stepper motor. Another challenge involves timing issues. Since the stepper motor is a physical device it has inertia and requires time to complete its steps. The student's code must allow sufficient delay between steps to allow for this motion to occur. Finally, it is possible that when the stepper motor reaches the special spot in its rotation that the photodetector will generate multiple interrupts when only one is desired. The student's code needs to be smart enough to ignore interrupts that occur in rapid succession. Students are asked to use the print routine developed for the round robin scheduler to print out how many interrupts were ignored as well as how many times the motor changed direction.

There are a number of different stepper motor setups, but one of the more exciting ones appears below [Fig. 3]. It functions the same as the setup described above but for added entertainment, a soda-pop can was mounted to the stepper motor axle. It is rewarding to students when they finally get their code working to see the can in action.

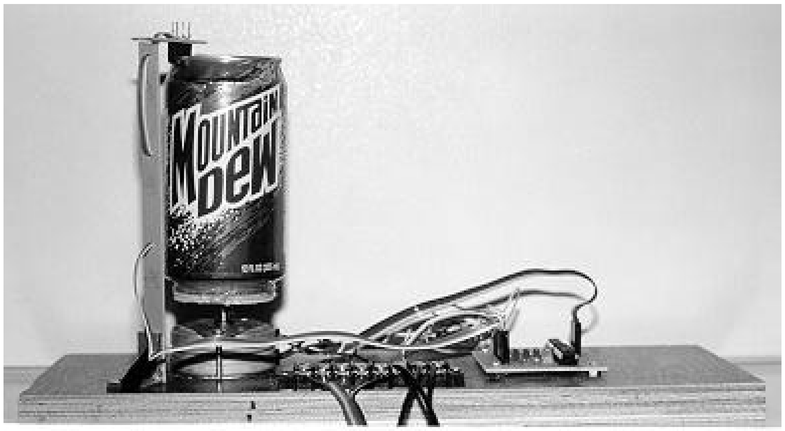


Figure 3: Soda-pop can version of stepper motor setup

Controlling an LED Display

Our EE course on Microcontroller Interfacing uses the MC68HC11EVB evaluation board. One assignment in this course requires students to write a program to control a four-digit LED display using two output ports of the 68HC11 microcontroller. The four-digit LED display attaches to the two ports via ribbon cables. The data on one port is used to control which LED is to be changed, while the data on the other port represents what gets displayed. The ports are controlled via memory mapped I/O, so a program can adjust the display by changing values stored in specific memory locations.

The challenging part of this assignment is much like that of the stepper motor assignment. First, because of the way things are wired, if the program running on the microcontroller placed a 7 on the port controlling the display value, the result displayed would not be a 7. Consequently students must figure out what values must be asserted on the ports to get the desired value. Second, there are timing issues. Programs written to drive the display must allow sufficient delay for the user to acknowledge a change. Without sufficient delay, a program driving the display as, say, a counter, would deliver a blurred display because the numbers were changing too fast.

A Sensor Driven DC Motor

Another interesting experiment uses DC motors and A/D converters. The DC motor setup attaches to an output port of the 68HC11. It uses a D/A converter that takes the digital value present on the output port and converts it to a voltage that controls the speed of the motor. In this experiment students are asked to apply a variable voltage source to the A/D converter input of the 68HC11 and then via a program interpret that voltage to generate a value that drives the motor. The variable voltage source that is inputted into the A/D converters is initially the output of an adjustable power supply. Once the setup is working, students move from using a power supply as input to the A/D converter to using a voltage-outputting sensor.

This assignment allows students to experiment with control laws, where the sensor is the input, the DC motor is the output, and the microcontroller program is the control. The program that drives the motor can be written to act as an amplifier, taking small changes in the sensor output and converting them to large changes in the motor output; or as a noise filter, ignoring rapid changes in input; or one of many other possibilities.

Using a Watchdog Timer

Another possible experiment involves performing tests using a Watchdog Timer. A Watchdog Timer is a safe guard available in many microcontrollers. It is designed to prevent uncontrolled infinite loops or suspended operation of the CPU. It is an interrupt mechanism that requires the microcontroller program to periodically check-in with the "Watchdog". If a program fails to check in after a certain amount of time the Watchdog Timer generates a non-maskable interrupt that forces the CPU to start executing a special recovery routine.

An experiment that can be performed to demonstrate the operation of a Watchdog Timer is to write a microcontroller program that performs some visible task, like driving a stepper motor, and then initiate the Watchdog Timer and have the program purposely fail to check-in. The results of this experiment would be that the stepper motor would turn for a while and then suddenly stop when the Watchdog Timer sends the CPU to the recovery routine.

The Potential

There are many things that can be done once an Advanced Microcontroller Laboratory is setup. The following lists just a few.

Banks of Evaluation Boards

To satisfy the needs of larger classes a number of evaluation boards will typically be required. If many of the experiments in the course are not going to involve interfacing, these evaluation boards can be grouped together to minimize space. We refer to this arrangement as a *bank*. The bank is nothing more than a bunch of evaluation boards mounted on some surface with connections to power and the terminal server. Once constructed, the bank can be mounted on a wall or tucked away in a corner to conserve on facility space. An example of eight evaluation boards mounted in a bank arrangement appears below [Fig. 4].

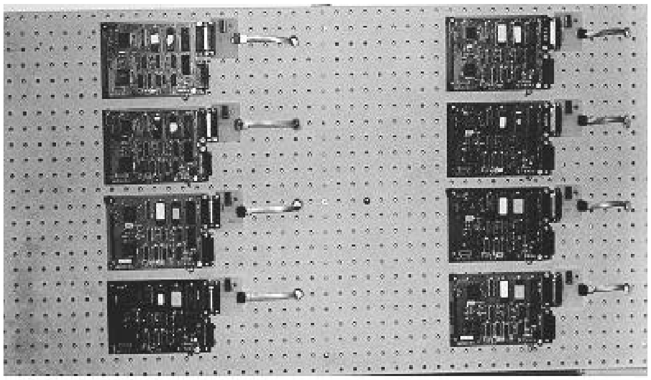


Figure 4: A bank of MC68HC11EVBs

A convenient variation used on the NDSU campus is to have a bank of evaluation boards in addition to a set organized in a lab-station arrangement. The bank is used for servicing the bulk of microcontroller needs while the lab stationed evaluation boards exist for performing interfacing experiments. For the part of the course where no interfacing experiments are being done, the lab station evaluation boards can be included in the rotary so that they act as part of the bank.

Windowed Environments

Perhaps the greatest advantage offered by the Advanced Microcontroller Laboratory is the ability to interact with resources over the network. This advantage can be further exploited by the uses of windowed environments such as X-Window or MS Windows. With the advent of falling computer prices and free UNIX (e.g. FreeBSD or Linux) running X-window, it becomes very affordable to set up an X-station such that a student can have a connection to the campus host in one window, a connection to an evaluation board in another window, a debugger or simulator running in a third window, and a text editor working on the lab report in a fourth window. With multiple windows students have the ultimate lab station. And with network-wired resident halls becoming more common, all this could be available from the student's room.

Distance Education

One of the biggest potentials that the Advanced Microcontroller Laboratory offers is in the area of distance education. A state or educational community will typically have multiple institutions that are spread out over a large geographic area. It is often desirable to offer similar courses at several institutions, and one means of doing this is through interactive video links via satellite. With this arrangement an instructor may be at one university teaching to a class with their lecture being simultaneously broadcast to other institutions where their students are watching via video monitors. This may work for the lectures, but it doesn't provide means for the lab component. In an ideal situation each institution receiving the lecture would have its own lab equipment for students to practice on. However, this isn't always feasible. In such cases the Advanced Microcontroller Laboratory would provide an excellent solution. Provided enough evaluation boards were present, it would be possible for students to connect to laboratory equipment hosted at another institution via the Internet.

Special Needs

The remote login ability of the Advanced Microcontroller Laboratory can make laboratory work easier for the disabled, students older than average, students with children, and anyone else with special needs that may have difficult making it to campus to perform exercises in a lab. Also during adverse conditions when travel is abnormally difficult this advanced lab can be a great convenience.

Student Feedback

After using the Advanced Microcontroller Laboratory for over a year, the results at NDSU have been positive. The laboratory setup has received favorable reviews from the students. In a survey of one of the classes, students were asked to rate the laboratory as either, not very convenient, satisfactory, or very convenient. Of 39 students responding, 28% said that it was satisfactory, 72% said it was very convenient, and none said it was, "not very convenient". In another class, 48 students responded to the survey with 29% claiming the lab was satisfactory, 67% saying it was very convenient, and only 4% saying it was, "not very convenient". One student commented, "[It's] more convenient being able to have access to everything from anywhere."

The Future

The Advanced Microcontroller Laboratory is undergoing further development. There is work being done on an interface that would allow students to remotely perform microcontroller interfacing experiments by allowing them to monitor and manipulate the states of the various ports on the microcontroller via the network. When this work is finished, this would provide a complete Microcontroller Laboratory on the Internet.

Conclusion

In conclusion, the Advanced Microcontroller Laboratory is a low cost solution to many of the problems associated with supporting a microcontroller lab. The simple, modular, and scalable design of the Advanced Microcontroller Laboratory allows it to be tailored to the needs of any microcontrollers or related course. It combines the best qualities of a simulator-based lab and an evaluation board based lab to provide superior access to equipment without hampering the types of experiments that are possible.

Using an Advanced Microcontroller Laboratory allows students to face the challenges of writing a round robin process scheduler, driving a pop-can mounted on a stepper motor, or controlling an LED display. And because of the potential of this advanced lab, students may perform these experiments using an inexpensive X-Window station that is housed in their room while accessing evaluation boards that are hosted hundreds of miles away.

Students who, in a survey, agreed that the lab was a very convenient way to access laboratory equipment have substantiated our claims that the Advanced Microcontroller Laboratory is a superior. Because of this, we feel that the Advanced Microcontroller Laboratory would be a worthwhile investment for any institution offering courses in microcontrollers.

References

[Greenfield 1992] Greenfield, J.D. (1992). The 68HC11 Microcontroller. USA: Saunders College Publishing

[Wakerly 1989] Wakerly, J.F. (1989) Microcomputer Architecture and Programming. USA: John Wiley & Sons, Inc.

[Burcin and Bohus, 1996]. Burcin, A., & Bohus, C., et al. (1996). Distance Learning Applied to Control Engineering Laboratories: IEEE Transactions on Education, 39 (3), 320-326.

[Infanger 1998] Infanger, B.D. (1998). Masters Thesis: A Microcontroller Laboratory on the Internet. Fargo, ND: North Dakota State University (in preparation)

Portable Computer Cluster Management

Cj Johnson
Technology Training Coordinator, Information Technology Services
North Dakota State University
United State of America
cjohnson@plains.nodak.edu

Stephen Stenehjem
Microcomputer Specialist, Agriculture Communication
North Dakota State University
United State of America
sstenehj@ndsuext.nodak.edu

So, you've just put a portable computer cluster together to deliver training anywhere, anytime, anyhow. But wait a minute! Who's going to keep track of the schedule? Who sets the monster up and takes it down? How do you manage the maze of wires, extension cords, and ethernet/modem connections? How do you move it from place to place? How do you make sure the information on the hard drives isidentical, and not containing “leftovers” from previous training sessions? And how do you handle obsolescence and maintenance? Join two people who've already made the mistakes with their portable clusters and they'll share some tips and tricks to make your first portable cluster a breeze to implement.

A Case-based Design for a Game Player

Paul Juell
Computer Science Department
North Dakota State University
juell@plains.nodak.edu

Patrick Paulson
Computer Science Department
North Dakota State University
ppaulson@plains.nodak.edu

This paper describes a design for a case-based game playing algorithm. Case-based systems make use of a comparison function to compare current situations with stored cases. This design uses a reinforcement signal that measures current performance to incrementally construct the comparison function. The comparison function is stored in a neural net which is trained using back-propagation. This method may be useful in other case-based application areas in which a deep understanding of underlying principles is lacking but where there is a measure of current system performance.

A Case-based Design for a Game Player

Paul Juell
Computer Science Department
North Dakota State University
juell@plains.nodak.edu

Patrick Paulson
Computer Science Department
North Dakota State University
ppaulson@plains.nodak.edu

Introduction

This paper describe a case-based game player that uses examples of moves executed in past games to determine the best move in the current situation. The player uses a library of board-positions that were used by the winning side in high quality games. At each turn, the player compares each possible move against board positions in the library. Since a library that contained all possible desirable board positions would be prohibitively large, a rule must is used to determine which move will result in a board position most similar to one that is recorded in the library.

The first section of this paper will present background material on case-based reasoning and reinforcement learning. Previous related work will also be described. The next section will present the design of the game-player. Then, results obtained from an implementation of the game player will be presented. The last section will discuss the results and avenues for future research.

Background

Case-based reasoning

Description

In *case based reasoning*, the current situation is compared against a library of cases. The most similar cases in the library are adapted to the current situation to provide a starting point to a solution for the problem at hand. A straight-forward case-based system for a game system, for example, would store the correct move to make for each possible board position. Then finding the right move for the current position is a matter of looking it up in the case library.

One of the advantages of case based reasoning is that it can be used in areas where there is not a strong theoretical model of the domain, such as economics, medicine, and law. [Kolodner 1993, page 27]. All that is needed to get started is a library of good (or bad) examples to serve as cases.

Case-based reasoning can also be helpful in domains were there is a strong theoretical model. Cases can be used to quickly propose solutions to complex problems, rather then solving the problems from scratch using domain knowledge. So, for example, rather then coming up with a new design for a building that meets designated criteria, a case library can be searched for other buildings that meet the criteria.

Shortcomings and difficulties

Indexing problem The indexing problem in case based reasoning consists of the *matching problem* and the *retrieval problem*[Kolodner 1993, pp. 19-20].

The matching problem is determining which features are important when selecting cases that match the current situation. These features may consist of surface features, which are readily apparent in the description of the case, and derived features, in which calculations are done using the surface features. This problem can be acute, considering that we want to use case-based methods in areas in which theoretical knowledge is limited [Ihrig & Kambhampati 1997].

The retrieval problem is how to organize the case data so that it can be searched quickly.

Adaptation After cases are retrieved from the case library, they may need to be modified so that they work in the current situation. Using game playing as an example, we might only store representative board positions and choose the case that best matches the current situation. If the move suggested by the case would not be legal in the current situation, adaptation would consist of determine which move would result in a similar configuration as that suggested by the retrieved case.

Reinforcement Learning

Description

In *reinforcement learning*, an agent learns behavior through interaction with an environment. The environment provides a reinforcement signal which indicates the desirability of the current behavior of the system [Kaelbling et al. 1996]. Reinforcement learning, also called *learning with a critic*, can be contrasted with supervised learning, or *learning with a teacher*, in which the environment indicates to the agent the correct action to take in a situation [Widrow et al. 1973]. Reinforcement learning is more difficult, since there is more information about correct outputs available in supervised learning. Reinforcement learning is also more applicable, since a supervised learning problem can be turned into a reinforcement learning problem by defining the reinforcement signal to be a scalar value derived from the difference between the current and the desired output. Reinforcement learning is also suitable in environments were many outputs may be equally valid. In this case the system can determine its own outputs rather than be required to learn some arbitrary set of outputs [Williams 1986]

An advantage of reinforcement learning in game-playing situations is that it is not necessary for the environment to know precisely which move the agent should make in any game situation. Rather, some measure of the the board position resulting from the agent's move can be used as reinforcement signal.

In some environments, there is more information available than just a reinforcement signal. In game-playing, for instance, an expert may be available to advise the agent on the correct moves to make. As mentioned above, reinforcement learning can still be used in such situations, but will take longer, since information will be lost in converting the information about the correct move to make into a scalar reinforcement value.

In the reinforcement learning model, at each step of interaction at time t, the agent is presented with a state x_t. In response, the agent produces an action a_t, which is passed to the environment. The environment responds with a reinforcement signal r_{t+1} along with a new state, x_{t+1} [Kaelbling et al. 1996].

Delayed reinforcement

One of the problems of using reinforcement learning is that the reinforcement signal for decisions made may not be available immediately. In game-playing, many moves may be made before it is known whether the game is won or lost. Once the reinforcement is received, there is the further difficulty of determining which of the preceding moves are responsible for the reinforcement. This has been termed the *credit assignment problem*. Sutton [Sutton 1984] has proposed the adaptive heuristic critic (AHC) algorithm to solve these problems. In this algorithm, a *critic* analyzes the delayed reinforcement and uses it to produce an estimate of an ideal reinforcement signal—one which produces a positive reinforcement when the agent performs a better than average action given the current context, and a negative reinforcement when the action is worse than average.

This estimate is produced by using *temporal difference methods*. In temporal difference methods, a reinforcement signal is calculated by determining the difference between temporally successive predictions of the outcome reinforcement to be received [Sutton 1988].

In the AHC algorithm, the heuristic reinforcement signal is calculated by keeping a vector of weights, with each weight corresponding to an input component. This vector is used to determine a *prediction of reinforcement forthcoming after time t* made at time s with the equation

$$p_t^s = \sum_{i=1}^{n} v_i^s x_i^t \tag{1}$$

The vector of weights is updated with the equation

$$v_i^{t+1} = v_i^t + \beta \hat{r}^{t+1} \bar{x}_i^t, v_i^0 = 0 \tag{2}$$

for $1 \leq i \leq n$ and $t = 0, 1, \ldots$. In this equation, $\bar{x}_i^t$ is a *trace*, or weighted average, of the values of x_i, with the more recent values weighted more heavily. This update rule causes weights to be increased if their corresponding inputs have contributed to p_t^s in the recent past and the heuristic reinforcement value is positive. If the value is negative, the weights are decreased for those components that have contributed to the prediction of rewards.

The prediction calculated in equation 1 along with the external reward signal is used to calculate the heuristic reinforcement signal with the rule

$$\hat{r}^{t+1} = r^{t+1} + \gamma p_{t+1}^t - p_t^t, \tag{3}$$

where $\gamma, 0 \leq \gamma \leq 1$, is scalar *discount rate parameter*.

Also kept is a vector of *traces* of the input values, with each component containing a weighted average of the successive values of the input component, with the more recent values receiving higher weight.

Connectionist implementations

There have been several connectionist implementations of the AHC algorithm. These include [Lin 1993] and those discussed in [Williams 1988] and [Williams 1986]. These approaches use multi-layer neural nets and back propagation in an effort to generalize the expected reinforcement of one input to similar inputs. This is the approach taken in the current design.

Related work

There has been much research in the development of good similarity functions for classifying research. In [Ihrig & Kambhampati 1997], cases are broken into component parts. These parts are retrieved to create plans to solve subgoals of new problems. The new plan has a set of constraints and actions designed to meet those constraints. The retrieval strategy is altered when the new plan fails, that is, when it has an inconsistent set of constraints.

Much research has been done in the selection of which features to include when evaluating similarity. [Wettschereck & Mohri 1997] and [Aha & Goldstone 1990] both present surveys of approaches to this problem. [Skalak 1994] discusses a genetic technique called *random mutation hill-climbing* to select features. [Caruana & Freitag 1994] uses hill-climbing methods to select attributes to use for classification using decision trees. [Liu & Setiono 1996] uses a probabilistic approach to feature selection. [Vafaie & Jong 1993] uses a genetic approach.

[Aha & Goldstone 1990] and [Scherf & Brauer 1997] both describe methods for assigning weights to features depending on their importance in discriminating which class instances belong to.

[Ricci & Avesan 1995] describes another technique for altering a similarity metric using reinforcement learning. This system adjusts two sets of weights using reinforcement learning. A *local* similarity function is used, which depends on the point in the input space from which the distance is taken. This allows conditions to be expressed such as “if feature A is greater than 70% then use only feature B and C to compute similarity”. The use of multi-layer neural nets in the current design should also allow such conditions to be expressed.

[Callan et al. 1991] describes *CABOT*, an approach similar to the one described in this paper applied to the game Othello. The strategies differ several ways. In CABOT, the case base contains instances of moves available. The program attempts to find a move that is similar to a successor of the current state. This is the same approached used in this design. In CABOT, however, if an exact match is not found, the found case is examined to determine what kind of transformation is required to get a board similar to the found case. Then all legal successors are examined to find the one that results in the most similar transformation. CABOT can then try to improve its performance by either changing the criteria for selecting cases or by changing how it selects a successor state that results in the most similar situation. In the current approach, these complications are avoided. The successor state that is most similar to any case in the case base is chosen. The algorithms also differ in that CABOT does not use a multi-layer network of weights.

Design of program

This section presents a design of a program that will learn how to play a game using case based reasoning. For any interesting game, there will not be enough storage for all possible board positions. If there is a library of good board positions, however, these can be used as evidence to help determine which of several possible moves is best.

Requirements for the Game

We assume the game implementation can provide, for each game board

1. a list of possible moves that can be made.
2. a set of real-valued features that describe the board.
3. an indication of when the game is over
4. an indication of the victor of the game if it wasn’t a draw.

Outline of approach

The player makes use of a database of board positions that are from games that were won.

When it is its turn to play, the player examines all possible plays it can make. For each board generated by these plays, it examples all cases it in its case library. The features are extracted from the generated board position and the case from the database. These features are used as arguments to a similarity function. The similarity function determines how different the two boards are.

After all the comparisons are completed, the minimum difference each potential move has with some case in the library is used to derive a probability that the move will be chosen: the higher the minimum difference, the lower the probability. The next move is chosen using these probabilities.

Similarity function

The algorithm uses an incrementally constructed similarity function. The results of play are used to construct a function that results in the most wins and fewest losses.

A multi-layer neural net is used to implement the similarity function. This is approach is used so that any learning that occurs might be generalized to similar inputs [Lin 1992]. The input to the neural net is the feature vectors of the two board positions to be compared. The output is a measure of the difference between the two board positions. This neural net is trained using back-propagation with the difference between its current output and an improved measure of the difference between the two boards used as an error measure. The improved measure is based on the results of current play.

If we had a way to determine the quality of board positions, then we could determine if the board positions were similar if they had close to the same quality. So boards that have close to the same quality should be similar. We can measure the difference in quality between between two boards as

$$\sigma^2 = (r_b - r_c)^2, \tag{4}$$

where

σ^2 is the difference between the proposed board and the case from the library,

r_b is the expected return starting from the proposed board, and

r_c is the expected return when starting from the case

The difference between this value and the current output of the similarity network can be used to train the network using back-propagation.

Return function

One way to create a return function would be to use known heuristics to evaluate the boards. But since this design is intended to be generally applicable to weak-theorem domains, a different approach is used. This approach will make use of temporal difference learning to incrementally develop a return function. A neural net trained with back-propagation is used to implement the return function.

After each move, a prediction is made about the return of the current board using the current return function. This prediction is compared to the prediction made for the previous board. The difference between the two predictions is used as an error term to train the network for the previous board.

Experimentation

When playing a consistent opponent, there is a risk that learning will stop unless the program behaves in a stochastic manner in order to allow new cases to be generated for comparison to the current cases.

As described above, each potential move is compared with the case library to find the case with the lowest difference in quality. But rather than deterministically choosing the move that has the lowest difference with some case in the library, the difference values are used to influence the probabilities that a particular move will be chosen.

For all potential moves m, let

$$V_m = \min_{c \in caselibrary} V(c, m) \tag{5}$$

The algorithm chooses move m with a probability inversely proportional to the ratio of V_m to $\sum_{allpotentialmovesn} V_n$.

Testing the design

The implementation

The design is currently being tested using an implementation of Tic-Tac-Toe. In the trials, the design plays matches with a Min-Max tree search algorithm which does a full ply search on the Tic-Tac-Toe board. At the end of each game, the board positions used by the winner are stored as cases.

The two players take turns going first. Different return and similarity networks are used depending on whether the player goes first or second.

Conclusion and future research

Problems with design

Search time

The program examines each case in the database to find the one that provides the best match. If there are a lot of cases this could be a bottle neck. The current program continuously adds new cases to the database as it plays games, and so the database could theoretically grow until all possible board positions are recorded.

There are several approaches that could remove this bottleneck

Fixed set of cases The set of cases could be fixed beforehand to include only boards that some expert has concluded are ’representational’ of all possible boards.

Generalization The program could only use a small subset of cases when it is searching for moves. The cases in the subset would be selected such that none of them were similar to each other using the current similarity function. As the similarity function changed, a new subset could be drawn from the set of all recorded cases.

Parallel search Several processors could be used to search a large case library.

Topology of neural nets

The networks used for generating the heuristic value and for comparing a potential move and a case are each trying to model a complex nonlinear function. In trials it has been difficult to determine a neural net topology that allows these functions to be modeled accurately with acceptable training times.

Future directions

Besides addressing problems discussed in the previous section, other improvements might be made to the game player.

'Equal' cases

If the feature sets of a move and a case from the library are 'equal', that his, have the same representation, compared bitwise, then either choose that move can be chosen as the board with most similarity. This might speed learning of the similarity rule, since the network used to implement the rule would not need to be trained to return no variance when the two boards used as input have the same feature values.

Use of a non-zero lambda

The temporal difference method used to generate the evaluation function is one of a class of methods called TD(λ) methods.

λ is a value, $0 \leq \lambda \leq 1$, which controls how much influence future reward, as opposed to immediate reward, has over the evaluation of the return from a game board. Let $\hat{V}_t(x)$ be the estimate of return for board x at time t. Then the TD(λ) return can be written recursively as

$$r_t^\lambda = r_t + \gamma(1-\lambda)\hat{V}_t x_{t+1} + \gamma\lambda r_{t+1}^\lambda \tag{6}$$

In the current implementation, a λ value of 0 is used. This means that only the estimate of return for the next time step is used to alter the estimate for the current time step. The use of a non-zero λ allow estimates of return from farther into the future to be used to alter the current estimate of return. Empirical studies have shown that non-zero λ can lead to increased learning rates [Peng & Williams 1994][Sutton 1988].

Use of an Expert

If experts are available on the game being implemented, their advice could be used to speed the construction of the similarity metric. Rather that using a heuristic rule to determine how different two boards are, the experts could be asked directly for a measure of the different results expected from two boards, and this value used to train the similarity function neural net. The experts' role is a simplification of the role played by them in other case based systems, in that they only have to come up with a value. There is no need for them to justify their value with a rule relating to the feature spaces of the two boards.

Conclusion

Since the design presented in this paper makes minimum use of game-specific knowledge, it is a *weak problem solver* [Luger & Stubblefield 1993]. This makes it useful for a wide range of gaming applications. However, this flexibility comes at the cost of very slow learning. Learning of some games maybe impossible without time-consuming experimentation with different network topologies.

If these problems can be solved, a unique approach to the incremental building of similarity functions in case-based systems will have been demonstrated. The technique of using reinforcement learning to perfect the similarity function may have wide applicability to case-based systems.

References

[Aha & Goldstone 1990] Aha, David W. & Goldstone, Robert L. (1990). Learning attribute relevance in context in instance-based learning algorithms. Program of the Twelfth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum Associates, 141–148.

[Callan et al. 1991] Callan, James P., Fawcett, Tom E., & Rissland, Edwina L. (1991). Cabot: An adaptive approach to case-based search, 1991, Proceedings of the Twelfth International Conference on Artificial Intelligence, International Joint Conference on Artificial Intelligence, San Mateo, CA. 803–808.

[Caruana & Freitag 1994] Caruana, Rich & Freitag, Dayne (1994). Greedy attribute selection. Proceedings of the Eleventh International Conference on Machine Learning. New Brunswick, NJ: Morgan Kaufmann.

[Ihrig & Kambhampati 1997] Ihrig, L.H. & Kambhampati, S. (1997). Storing and Indexing Plan Derivations through Explanation-based Analysis of Retrieval Failures. Journal of Artificial Intelligence Research, 161–198.

[Kaelbling et al. 1996] Kaelbling, Leslie Pack, Littman, Michael L., & Moore, Andrew W. (1996). Reinforcement learning: A survey. Journal of Artificial Intelligence Research, 4, 237–285.

[Kolodner 1993] Kolodner, Janet (1993). Case-based reasoning. San Mateo, CA: Morgan Kaufmann Publishers, Inc.

[Lin 1993] Lin, L.-J. (1993). Reinforcement learning for robots using neural networks. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

[Lin 1992] Lin, Long-ji (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine Learning, 8, 293 – 231.

[Liu & Setiono 1996] Liu, Haun & Setiono, Rudy (1996). Feature selection and classification - a probabilistic wrapper approach, 1996, Proceedings of the Ninth International Conference on Industrial and Engineering Applications of AI and ES.

[Luger & Stubblefield 1993] Luger, George F. & Stubblefield, William A. (1993). Artifical Intelligence. Redwood City, CA: Benjamin/Cummings, second edition.

[Peng & Williams 1994] Peng, J & Williams, R. J. (1994). Incremental multi-step q-learning. Proceedings of the Eleventh International Conference on Machine Learning. San Francisco, CA: Morgan Kaufmann, 226–232.

[Ricci & Avesan 1995] Ricci, Francesco & Avesan, Paolo (1995). Learning a local similarity metric for case-based reasoning. Proceedings of the First International Conference on Case-Based Reasoning. Sesimbra, Portugal: Springer-Verlag, 301–312.

[Scherf & Brauer 1997] Scherf, M. & Brauer, W (1997). Feature selection by means of a feature weighting approach. Technical Report FK1-221-97, Institut für Informatik, Technische Universistät München.

[Skalak 1994] Skalak, David B. (1994). Prototype and feature selection by sampling and random mutation hill climbing algorithms, 1994, Proceedings of the Eleventh International Machine Learning Conference, New Brunswick, NJ: Morgan Kaufmann. 293–301.

[Sutton 1988] Sutton, Richard S. (1988). Learning to predict by the methods of temporal differences. Machine Learning, 3, 9 – 44.

[Sutton 1984] Sutton, R.S. (1984). Temporal Credit Assignment in Reinforcement Learning. Ph.D. thesis, University of Massachusetts, Department of Computer and Information Science.

[Vafaie & Jong 1993] Vafaie, Haleh & Jong, Kenneth De (1993). Robust Feature Selection Algorithms, 1993, Proceedings of the 5th IEEE International Conference on Tools for Artificial Intelligence. 356–363.

[Wettschereck & Mohri 1997] Wettschereck, D. Aha, D. W. & Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. Artificial Intelligence Review, 11, 273–314.

[Widrow et al. 1973] Widrow, B., Gupta, N.K., & Maitra, S. (1973). Punish/reward: Learning with a critic in adaptive threshold systems. IEEE Transactions on Systems, Man, and Cybernetics, 455–465.

[Williams 1986] Williams, R. J. (1986). Reinforcement learning in connectionist networks: A mathematical analysis. technical report ics 8605. Technical report, Institute for Cognitive Science, University of California at San Diego, La Jolla, CA.

[Williams 1988] Williams, R. J. (1988). On the use of backpropagation in associative reinforcement learning, 1988, Proceedings of the IEEE International Conference on Neural Networks, Institute of Electrical and Electronics Engineers, New York. I–263–I–270.

Converting a Course's Presentations from Legacy Material to the Web

Paul Juell
Computer Science Department
North Dakota State University
juell@plains.nodak.edu

Abstract

This paper describes some of the process of converting a class from legacy material to Web based presentations. Previously the lectures for this course were presented using transparencies and an overhead projector. The overhead projector has been replaced by a computer running a Web browser, and a high quality computer projection system. The legacy material, in this case, was primarily 312 transparencies stored in 3 ring binders. This conversion process followed 4 steps: placing the legacy material online, indexing and structuring the legacy material, building the new lecture structure, and adding new material to the lecture material.

Introduction

This paper describes some of the process of converting a class from legacy material to Web based presentations. Previously the lectures for this course were presented using transparencies and an overhead projector. The overhead projector has now been replaced by a computer running a Web browser and a high quality computer projection system. The legacy material, in this case, was primarily 312 transparencies stored in 3 ring binders. This conversion process followed 4 steps: placing the legacy material online, indexing and structuring the legacy material, building the new lecture structure, and adding new material to the lecture material.

The Course

The particular course described here is CS730 - Office Information Systems. This course includes evaluation of technologies to be used in an office setting. The author has taught this course every few years since 1981. In that time he has developed a number of transparencies, files, tests, slide shows and other material for the course. Over that time, there has not been a book that fully supports what the author was trying to do in the class. This has meant that the students have had more need to get access to the lecture materials. In the past, copies of the previous years material have been made available for copying. This has worked reasonably, but does not give the students the new material developed after they have gotten their copy of the material. In addition, this process has always been a logistical problem.

The course was taught in a computer cluster room, with each student at a workstation. The material projected was also available at the student's workstation. The room had a printer, which the students often would use to obtain current copies of the lecture material. The last time the class was taught, all lecture and student presentations were projected web pages, shown directly from the browser. The author used a radio mouse (LOGiTECH TrackMan(TM)Live). This allowed him to pace the room while presenting the lecture.

The Process

The basic process followed had four major parts:

- scan the legacy material
- index and structure the on-line material
- building the structure of the lectures
- repackaging and augmenting the lecture material

Two major factors that shaped and drove the process were the sheer volume of the legacy materials and the day to day deadlines of having a lecture ready. The basic hope was that the availability of the legacy material would limit the time required to prepare the day to day presentations. While getting all of the legacy material on-line helped there were still several time consuming steps to getting the material in a form that could be used for lecture. In some cases, the lecture was not fully setup in a final form, and pages were selected out of sequence from the master index into the legacy material. Most of the time, however, the lectures were fully prepackaged and could be presented and followed by clicking on the hot links in the order displayed.

Scanning the Legacy Material

The legacy material consisted of a collection of transparencies, 3 slide shows, computer files, some sources and items to be passed around the class. The primary part of this collection was a set of 312 transparencies. These where three hole punched and stored in 3 ring binders. There were in two groups, a group in the order presented in the last class and a somewhat chaotically ordered group of old and new transparencies that were not used in the last class. The process of scanning the transparencies took large blocks of time. For this course it took about 30 hours. The first time a reasonable amount of time was taken in finding a good size (we used 560 by 720 pixels), finding settings that would produce good copies and some other startup type problems. I have done this for two other classes and it has taken from 13 to 20 hours for each class. It important to have a unique name for the image of each transparency and to maintain the order of parts of the collection. It is also important to not create too much clerical bookkeeping, as this makes it harder to start and stop the process and to farm it out to other people to do the scanning. The solution the author used was the following naming scheme. Each file was named in the pattern: fMMDDNN. The date the scanning was performed was coded in the MMDD. MM was the two digit number of the month and DD the two digit number of the day of the month. NN was the number of the image scanned that day. If you started and restarted on a day, you would either take the next value of NN or some number larger than previously used. This gave a large collection of image files with unique names and the names would list in ascending order. Remember to code the zeros or you will not get the correct ordering of the lists.

Attempts were made to use OCR on the transparencies, and this was a complete failure. It was hard to scan the transparencies. But, the transparencies were the result of multi-generation of copying through a copier. On other sets of material, not transparencies, not multi-generation copies, I had very good results in OCR scanning.

The slide shows (collections of 35mm slides) were convert into computer files by sending them to Kodak and getting a PhotoCD produced. This produced very high quality, color corrected images. Since then I have tried one of these "get your photo on a floppy" services (not Kodak) and was not happy, because the images were not color corrected. Since the first images were done, NDSU has gotten a slide scanner, and that has worked very well. Because I have a relatively small number of slides, a simpler naming system was used, the files were names sGGNN, where GG was the group number and NN was the number within the group.

The items to be shown to the class were photographed with a Polaroid camera, and the pictures were then scanned. Since this was normally just one or two items at a time, meaningful names were given to the image and the images were placed with the lecture material.

A few of the items of paper were also scanned in, and added to the collection of transparency images. The rest of the legacy material were already in computer files. These were placed in directory and a hot link was pointed at the directory. At some future time, these files may be converted to html documents and a reasonable indexing structure may be built.

At this point an archive copy of the raw computer material was made. NDSU has a CDROM burner, so a CDROM was burned of all of these files. This backup at this point has been used several times, because the a number of files have been lost or damaged in the process of building the presentations. The backup has allow speedy fixing of the problems when time has been critical.

Structuring the Legacy Material

There must be some type of structured view of information to deal with this collection of several hundred files. The first attempt to provide this structure had severe problems. The plan was to build a number directories representing concepts and move the appropriate files into these directories. Then build indexing structures pointing into the directories and to the files.

The author found this process unworkable for two reasons. One, the process of moving the files around, finding the files, and other simple processors were slow and error prone. Once the files were moved they were hard to find for use in other contexts. The second problems was that the tools to look at the images basically allowed a person to deal with one image at a time. Some of the tools did have a way to show a collection of thumbnails of the files, however, these were small thumbnails (from about 30 by 30 to 60 by 60 pixels). In these small views, almost all of the transparencies looked identical. The small thumbnails are fine for picking out pictures with different colors and objects. However, most of the transparencies were just a block of text, and so looked similar with these small images. A number of things were tried, but the one that worked best was to develop large thumbnails and automatically build Web pages indexing these super-sized thumbnails.

Shell scripts were developed that went through all the files and produces 25% sized thumbnails for each file. The thumbnail was named the same as the original file with s- at the front of the name. The scripts also produces a file named all.html which showed all of the thumbnails in name order. Each thumbnail was hot linked to the original file.

The full all.html page was displayed and inspected. Concepts were determined by inspecting the thumbnails and visiting some of the full sized images. For example, a group of images may discuss bar codes. An editor would be used to create a file bar-codes.html. The file all.html would be copied into bar-codes.html and the lines referring to images about bar codes would be keep and the rest would be deleted. This continued until most of the files had been classified.

Structuring the Presentation Material

It was intended to replace the legacy material with new, native HTML documents. To make this conversion easier a separate directory was created for the lecture material. All new material was placed in the lecture directory.

Legacy material was referenced by pointers back to the html pages built in the legacy directory. All lectures, for the whole term, were rooted in a page called lecture.html. This page is basically an outline of the class. Each entry on the page points to transparencies and Web pages that represent several days of lectures.

The starting version of the lecture page pointed to the index pages created for the legacy material. As the course progressed, the new lectures were added to the lecture page some time before the class presentation.

Some times material was accessed out of the pre-planned order. If the material could not be found in the prepackaged lectures, the all.html page was used. This and other material that had not yet found a home in the lectures were pointed to from a special section at the bottom of the lecture.html page. This allowed finding other resources during a lecture.

Converting to New Pages and Augmenting the Material

If the legacy material was used without change, the lecture.html page would point to the topic index page in the legacy material. If this was augmented or updated, the index page was read in using Netscape Gold with the flags set to update the pointers for remote access. The page was then edited and placed in the lecture directory. When there was time, topic pages were recoded into html format. The author normally works in UNIX, so the tools included xfig. The drawings were redone in xfig and exported to gif files. A 25% sized image was created using xv, and this image was included in-line in the topic page and were hot linked to the full sized image. When tables were created, this was done in Netscape Gold. The author did the rest of the editing directly in the vi editor. When sections were augmented or replaced, the author found he usually created significantly more information than he has in the previous transparencies. Since the new files represented concepts, rather than just a number images, they tended to receive meaningful and descriptive names. Since small collections of new pages typically represented a large number of old pages, there has not been a problem with too many pages to understand what items are there.

The Classroom and Student Interaction

For most classes, the author could walk in and click on the hot links in order and present the material. The extra pointers were available, which allowed using the legacy materials when there was not time to fully get the lecture packaged into the lecture page before class. For many classes, the students had printed the day's lecture from the web before class and were taking notes directly onto the copies of the lecture material. One time, a review sheet was ready about 1 and a half before class, however everyone already had copies at class time. Having all of the material on-line gave more flexibility in responding to student questions. Since the projector and web was used by the instructor and was available. When the students gave their presentations, they simply did the same with a Web based presentation.

Updating After the lecture

Some times there were problems found in the presentation material during the lecture. Normally corrections would be made soon after class, and the students would go and get the corrected versions. In addition, students would e-mail about problems, and then these would be corrected. Normally a large note would be placed on the class home page if the changes were major. This has allowed the students access to timely material.

Summary of Results

The process of converting legacy material does take time, but can be set up to be a straightforward routine. This routine has not been too expensive in time. This process has allowed the author to change and update material, often easier that it would have been using the physical transparencies. In addition, it allows the student easy and timely access to the actual materials used in the class lecture.

"Year Three of NDSU's Instructional Web Project"

Dr. Paul Juell
Associate Professor, Computer Science
North Dakota State University
United States of America
juell@plains.nodak.edu

Dr. James B. Ross
Associate Director, Information Technology Services (Learning Technologies)
Interim Director, Center for Academic Information Technology
North Dakota State University
United States of America
ross@plains.nodak.edu

Abstract

For two years, the innovative faculty who directed NDSU's Instructional Web Project (IWP) extolled the benefits of supplementing their courses through WWW development. Now, in its third year, the NDSU Instructional Web Project has undergone significant change. WWW development has become an institutional imperative rather than an experiment on the "bleeding edge." One of the driving forces of that change is heavy reliance by faculty on using the web in conjunction with courses. Using the web to deliver and retrieve information is now second nature for a large number of NDSU faculty and most of their students. We will discuss recent institutional initiatives for supporting faculty who work on the web and the consequences of these new initiatives on the project. Use of instructional technology has exceeded the original expectations of IWP and shows every sign of continuing this trend.

"Year Three of NDSU's Instructional Web Project"

Dr. Paul Juell
Associate Professor, Computer Science
North Dakota State University
United States of America
juell@plains.nodak.edu

Dr. James B. Ross
Associate Director, Information Technology Services (Learning Technologies)
Interim Director, Center for Academic Information Technology
North Dakota State University
United States of America
ross@plains.nodak.edu

Institutional Initiatives

NDSU President Thomas Plough helped set the stage for year three of NDSU's Instructional Web Project (IWP) in his 1996 "State of the University Address":

> Our students are visual learners, accustomed to highly sophisticated mass media productions. The traditional lecture/discussion mode of teaching just does not match up well with our students' learning styles. Multimedia classroom presentations are coming and our faculty will have to adopt and adapt to this powerful new delivery system. But use of multimedia is not easy. The learning curve is steep, even for bright and technologically literate faculty. So training and support are important.[1]

One institutional response to the need for faculty training and support came from the Information Technology Roundtable (ITR) last year. The President charged the ITR to "integrate technology into the infrastructure of NDSU," and the ITR's "Working on the Web" subcommittee (WOW) noted that WWW development "resources are spread across several campus departments [and service units] with little tying them together. A new structure, singularly dedicated to this outcome, is called for."[2] WOW established a one-stop shop for the creation of multimedia on campus, called the Center for Academic Information Technology (CAIT). The biggest demand from faculty, staff and students is, of course, for WWW development, and the CAIT took over the IWP in its third year. The Interim Director of the CAIT hired an Instructional Designer and a Multimedia Technologist for direct support of instructional efforts.

Project Objectives

Historical Perspective

The objectives of the IWP have changed over time. In its first year, each of the eight faculty participants merely wanted to put one of their courses online. Each of them did this during the Fall Semester 1995, but none could stop developing and most added one or two courses during the Spring of 1996. Other faculty joined the effort that spring, making the experiment a full-blown, grassroots effort to integrate technology in the curriculum. The membership in IWP had grown to twelve active participants, but the web effort extended to sixteen departments and forty-five courses. It seemed time create a master plan, the IWP sought funding for more servers, more student assistants, and each volunteered to lend five hours of their time each week over the course of the next year as WWW evangelists, contacting departments and faculty and encouraging them to create a web presence. The faculty worked closely with Information Technology Services designing and implementing a new, more robust and high-performance WWW server system which went into production the following Spring. They interested over half of the departments in WWW development and, as a direct result of the student and staff support that they afforded the campus, thirty-one departments hosted 111 WWW enhanced course in Spring 1997.[3]

Instructional technology efforts are becoming a very significant aspect of more and more courses at NDSU. The IWP deserves a great deal of the credit for changing the way the campus works on the web. Many faculty are developing electronic materials for their courses which allow students to access course materials on campus or at a distance. Other faculty members are developing multimedia modules that students access directly from a workstation on campus via the WWW. In the Spring of 1997, IWP began supporting a "Student Academic Web Server," to accommodate professors who assign students individual and group projects and research and require students to present their findings on the web or as multimedia in the classroom. Currently fifty-four individuals are using this server to exchange course related information.[4]

Under the direction of the Instructional Web Project and the CAIT, use of the WWW soared. Figure 1 and Table 1 document the rise of WWW enhanced courses indexed for NDSU users first by the IWP and now by CAIT:

Table 1.

Semester	Courses
Fall 95	11
Spring 96	45
Fall 96	103
Spring 97	111
Fall 97	175
Spring 98	156

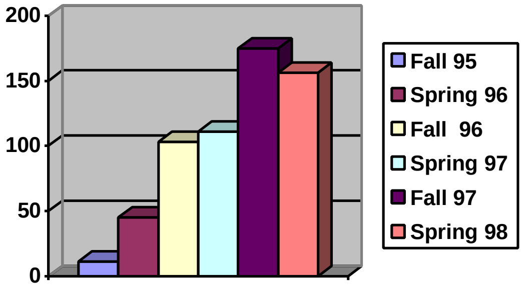


Figure 1.

The permanent CAIT staff cannot carry out the charges of the CAIT alone, so just as the IWP faculty did we employ undergraduate and graduate student WWW experts. These students work with the permanent staff to develop electronic materials for faculty, staff and other students. This team approach has resulted in more sophisticated and effective WWW sites and multimedia projects than individuals can develop working alone. The CAIT promotes a process that brings students, staff and faculty together as productive teams under the management of the Instructional Designer, Multimedia Technologist, NDSU Webmaster and the Multimedia Center Coordinator. The projects are done outside the classroom, to foster new relationships between students and faculty. We produce a different classroom environment using the work that students and faculty undertake collaboratively outside of the classroom.

The CAIT staff, including student employees, and the faculty work together to identify the learning objectives on which they will bring technology to bear. The faculty then work with the Instructional Designer and Multimedia Technologist to make flow-charts of the necessary tasks, plan the application of multimedia using storyboards, and set deadlines and timelines. CAIT staff assign to the faculty and student employees the necessary duties for design, development and evaluation. The faculty and students research and write scripts together, acquire media and copyright clearance, etc. In short, faculty, CAIT staff and students act as co-producers.

As year three progressed, a new phenomenon has begun to have profound consequences. Many faculty who have followed the lead of the Instructional Web Project faculty and begun to work on the web. They behave more like the IWP "early adopters" than the IWP faculty assumed was possible. These are not hesitant conservatives for whom all the benefits must be perfectly apparent before they try adopting new technology. They tend to pick up where their predecessors left off and use available technology extensively and intensively once they begin.

The IWP faculty never saw themselves as role models. Their colleagues lacked the IWP faculty's advanced WWW skills. As a consequence, the IWP employed and trained students with the objective of offering the student experts as resources to other faculty. The students were to provide services to the campus which flattened the learning curve and lowered the barriers to working on the web. Most faculty who consult with CAIT staff, however, clamor for the latest and greatest WWW technology and seem less concerned with the technical barriers than we thought they would be. They are finished waiting and want to charge ahead. Obviously, this represents a pending support crisis for those who work with and advocate for the integration of technology in the curriculum. The old model of assigning an individual student to a department or faculty member for several hours to get a new effort off the ground no longer meets the demand.

We have begun to develop some new strategies to meet this increased demand for web services and instructional support. A new faculty development program, called Academic Technology Partnerships (ATP), invites faculty who want to add cutting edge web technology to their courses to work in the CAIT during the summer. The CAIT staff are also building databases of instructional components which generate new WWW pages based on users' queries.

ATP

Faculty who join ATP are expected to develop high content/high interactivity course materials using the latest information technologies. ATP faculty have the CAIT at their disposal during the summer months. The Vice President of Academic Affairs selected eight faculty to participate in ATP this past summer and assigned them to the work 100% of their time and rewarded them with full summer stipends.

The direct outcomes related to the ATP instructional technology development efforts are:

Electronic course materials with rich content and a high degree of interactivity between the student and instructor, among students and between the student and the course materials that support improved student learning and demonstrate the role of NDSU as a leader in electronic course delivery.

A faculty with skills to improve course materials using the latest information technology tools that can be adapted to other aspects of the university.

Assessment techniques to determine the effectiveness of information technology upon improving student Learning.

Some of the most recent accomplishments of the CAIT staff have come as a result of trying to meet the new demand for instructional technology.

The most frequently used device this year is the Course WWW Page Generator. Faculty can use up to six components to build a WWW site for their course. The user chooses whether to construct an Objective and Main Index page, a Syllabus page, a Textbook and Related Links page, an Assignments and Projects page, an Exams and Quizzes page, and a Grades page simply by filling in forms in the WWW browser. These items are actually queries to a database. The professor supplies the course content by entering it in the WWW form and the database arranges it and produces HTML files for WWW publication.

The student employees and staff built custom devices with faculty to meet individual course needs. A prime example is a "Wallcovering Module" that helps interior design students estimate the amount of wallpaper needed to decorate a room. The interface is three dimensional, virtual reality, and students are given an interactive set of problems which begin with taking basic measurements and end with defining a budget for an interior design project. Each student's deliberations and progress is recorded in a database to which the professor has access.

The most ambitious ATP site is a Virtual Model United Nations (VMUN). NDSU is building an immersion environment that enables individuals across the country and around the world to exchange research and views about important global issues using the WWW for asynchronous learning and desktop video-conferencing for synchronous communication. NDSU is the host of the VMUN Security Council and we are partnering with other international universities whose faculty have agreed to host the other four branches of the UN.

Dynamic HTML

The CAIT is automating many of the tasks which were assigned to students working for the IWP. The CAIT will still construct web pages for faculty, for example, but we will not assign a student to work long hours on a professor's course pages if all the professor wants is the student's labor. The students will construct libraries of course web page templates from which faculty can chose, and then customize those templates to meet the faculty's requirements. Minutes rather than hours will be dedicated to these menial tasks once the templates are in place. The students that the Multimedia Center employs are too gifted for us to squander their talents on work that faculty could arrange to have done in other ways. It is, frankly, too expensive to employ students to work below their abilities. Likewise, all CAIT projects will be team projects in order to take advantage of the diverse talents that the student will bring to CAIT rather than assigning a whole project to one student. The CAIT is also responsible for keeping abreast of emerging technology and incorporating the best practices in learning technologies in the life and work of the campus. Consequently, CAIT student employees, as part of the team assignments, will investigate national and international trends in adopting, developing and deploying learning technologies.

The examples of dynamic HTML which follow reside on WWW servers which the CAIT has named "wow" after the subcommittee which established us as a university resource.

The Automatic Button Generator is a good example of embodying the skills of a designer in an application so that several objectives are achieved.   NDSU's WWW employs proprietary fonts in its navigational buttons and the buttons themselves are comprised of many "layers" in order to achieve a shadow effect. The font costs $150 for each workstation and using the fonts and effects to make each new button requires considerable Adobe Photoshop skill. Assigning students the task of filling orders for new navigational buttons would have depleted our valuable resource of student expertise and cost a great deal in software licenses for the font and for Photoshop. Instead, we purchased a WWW server, installed the font and Photoshop on it, and created scripts which fire up Photoshop when a user requests a new button. The request comes to the server via a database which contains "button components" -- that is, different scripts run based on the queries from the client. The server renders a new graphic image with the requested text on top of any available button style depending on client preferences. The client then saves the image for later use. We have reduced the necessary copies of the font and of Photoshop and recorded the steps required to render "Official NDSU buttons," thus freeing human and material resources for other purposes.

The Course Index is our most extensive and comprehensive effort to-date. The system allows professors to register with the server, receive a login name and password from the server, and to add, delete, update and replace links to their WWW instructional material. Students who use the database-driven utility receive only the information they seek, and the index is the number one referring WWW site to the main NDSU WWW. The index has also received the "Site of the Week" award by Everyware Development Inc., a firm which makes one of the products we use for dynamic web pages: Tango for Filemaker Pro 4.0.

Online Discussion Forums and Online Quizzes also supply needed tools for enhancing WWW instructional sites.

The CGI creator is another example. It dynamically creates a file that processes HTML forms. The CGI creator has two distinct advantages. First, it eliminates spending hours or days writing the script that processess HTML forms - the CGI Creator creates the script for you in a matter of seconds! Secondly, a tab-delimited text file is automatically created for you when you request your CGI file. The test file stores all the information submitted through the form.

Summary

NDSU has embraced WWW as a component of the delivery of instructional materials. For many faculty, staff and students, WWW has become the presentation tool of choice. The current IWP is still very active. Five IWP members have received a NSF grant to develop more sophisticated WWW applications. The rest of the campus is utilizing resources that IWP first requested and that the CAIT is managing.

[1] [Pres. Thomas Plough, (September 10, 1996).] "The Integrity of NDSU as the Land-Grant University of North Dakota," State of the University Address.

[2] [Report to the ITR from the WOW (Working On the Web) Subcommittee, approved by ITR January 30, 1997]"Proposal for the Future of Multimedia Technology at NDSU." See http://www.ndsu.nodak.edu/itr/wow.html

[3] See http://www.ndsu.nodak.edu/instruct/wwwinstr/cait/projects/WWWIC96.html

[4] For more detailed information on the IWP project, please see [Juell, et. al. (1997), Juell, Paul and others, "Year Two of NDSU's Instructional Web Project," 30TH Annual Small College Computing Symposium, PROCEEDINGS, pp. 205-209]; [Juell, et. al. (1996), Juell, Paul and others, "North Dakota State University's Instructional Web Project," 29TH Annual Small College Computing Symposium, PROCEEDINGS, pp. 550-555]; [Juell, et. al. (1996), Juell, Paul and others, "The Use of World Wide Web in Support of the Classroom," 29TH Annual Small College Computing Symposium, PROCEEDINGS, pp. 306-312]; and [Juell, et. al. (1997), Juell, Paul and others, "NDSU's New Web Server," 30TH Annual Small College Computing Symposium, PROCEEDINGS, pp. 199-204]

A Course on Genetic Algorithms for Seniors and Graduate Students

Bryant A. Julstrom
Department of Computer Science
St. Cloud State University
St. Cloud, Minnesota 56301 USA
julstrom@eeyore.stcloudstate.edu

Abstract

The genetic algorithm is a probabilistic population-based search algorithm inspired by the mechanisms of biological evolution: natural selection and genetic crossover and mutation. Though a recent development in computing, genetic algorithms have emerged as a powerful general technique for addressing problems of search and optimization. This paper introduces genetic algorithms, including a brief example, and describes a recently developed course over them: its prerequisites, content and organization, and assignments, as well as texts, journals, and other materials on genetic algorithms that can be used to support the course.

1 Introduction

The *genetic algorithm* (GA) is a probabilistic search procedure inspired by the mechanisms of biological evolution: natural selection and genetic mutation and recombination. A GA maintains a population of data structures, called *chromosomes*, each of which encodes a candidate solution to the target problem. Chromosomes are generally strings over some small, often binary, alphabet, but they may also be permutations of symbols or two-dimensional structures such as matrices or trees. The algorithm initializes its population with randomly generated chromosomes, and assigns each a fitness value that indicates the quality of the solution the chromosome represents.

The GA repeatedly selects chromosomes from the population to be *parents* in such a way that the more fit chromosomes are more likely to be chosen. This process imitates natural selection; more fit organisms have more reproductive opportunities. The GA applies operators inspired by biological genetic processes to generate new chromosomes, called *offspring*, from parents. *Crossover* imitates genetic recombination; it combines genetic material from two parents. *Mutation* randomly modifies a single chromosome. The GA evaluates each offspring chromosome these operators build to determine its fitness.

A *generational* GA generates enough offspring to form a new population, which then replaces the original population, and the process continues. As the generations succeed one another, the more fit members of the population propagate and recombine to build chromosomes of improved fitness, which represent improved solutions to the problem. The process continues until a stopping criterion is met, usually either that a fixed number of generations has passed or that the algorithm has identified a chromosome of suitably good fitness. When the algorithm terminates, the GA reports the best chromosome in the population. [Figure 1] summarizes the genetic algorithm.

An example GA illustrates this description. Consider the familiar *traveling salesman problem* (TSP), in which we are given a collection of n cities and the distances between them, and we seek a closed tour of the cities that visits each city exactly once in the shortest possible distance. A GA for this problem encodes tours as permutations of integers associated with the cities. The algorithm is

```
initialize the population with random chromosomes;
evaluate the chromosomes;
while ( stopping criterion not met )
  { select parent chromosomes;
    apply genetic operators to build offspring;
    replace existing chromosomes with offspring;
  }
report the best chromosome;
```

Figure 1: A summary of the genetic algorithm, which applies biologically inspired operators to a population of chromosomes, each of which represents a candidate solution to the target problem.

generational, with population size equal to five times the number of cities in the target TSP instance, and it preserves the best chromosome of each generation into the next.

[Figure 2] illustrates the GA's progress in a run on a well-known problem [Oliver, Smith, & Holland 1987], whose 30 cities are scattered on a 100 × 100 grid and whose shortest tour length is 420. The figure shows the best tour in the population in generations 1, 3, 10, and 21 of the run. In the first generation, the best tour is scrambled, with many crossing edges. By the third generation, the best chromosome shows some organization, with many long edges replaced by more direct ones, a trend that continues in generation 10. By generation 21, the GA has identified an optimal tour.

At St. Cloud State University, a course on genetic algorithms has been developed in the Department of Computer Science and offered three times in the last four years, including during the current quarter, under the department's Topics number, CSCI 475/575. The numbers indicate the target audience: seniors in a computing major and graduate students. This paper describes that course. In particular, the following sections outline the course's prerequisites, content, and assignments, and describe some resources available to support such a course: texts, journals, conference proceedings, on-line materials, and software.

2 Prerequisites

Genetic algorithms use interesting data structures (the chromosomes that encode candidate solutions, the population) and algorithms (parent selection, the genetic operators). In order to write, evaluate, and describe GAs, students must be fluent in a high-level programming language and have completed coursework in computer science at least through a course on data structures.

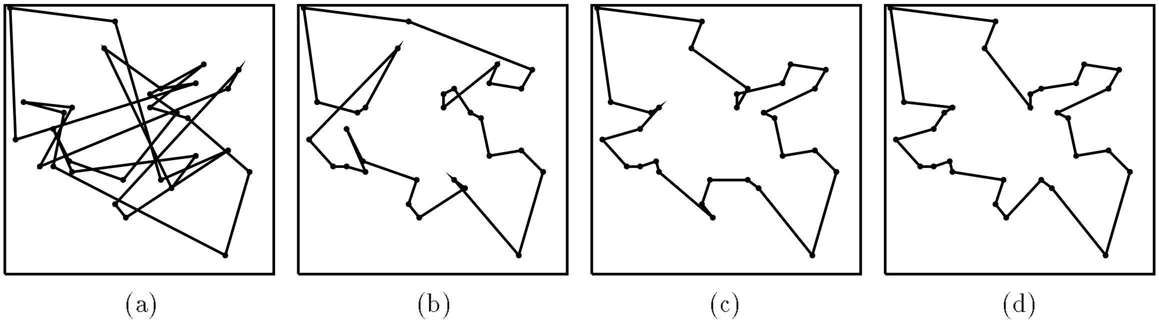


Figure 2: Best TSP tours represented in a GA's population in generations 1 (a), 3 (b), 10 (c), and 21 (d) in a run on a 30-city problem. The last tour is optimal; it has length 420.

3 Content

The course begins with a description of problems of numerical and combinatorial search and optimization, which genetic algorithms typically address, and with a summary of the principles of evolutionary biology and genetics that inspire GAs. This leads to a discussion of a GA's defining data structures—the chromosomes that encode candidate solutions and the population they comprise—and its essential operations: selection, crossover, and mutation.

A complete GA for a simple combinatorial problem illustrates these ideas. This GA is generational, encodes candidate solutions as binary strings, and chooses chromosomes to be parents according to probabilities that are proportional to the chromosomes' fitnesses, a scheme called *fitness-proportional selection*. A good problem for this demonstration is PARTITION: Given a list of positive integers $n_1, n_2, n_3, \ldots, n_k$, partition them into two sets $n_{i_1}, n_{i_2}, \ldots, n_{i_m}$ and $n_{j_1}, n_{j_2}, \ldots, n_{j_l}$ such that the difference between the sums of the integers in the two sets $\left|\sum_{a=1}^{m} n_{i_a} - \sum_{b=1}^{l} n_{j_b}\right|$ is minimized.

Another generational GA optimizes a real-valued function of several variables. The five functions of the De Jong test suite [De Jong 1975, Goldberg 1989] are good examples here; they have long been associated with GAs and appear frequently in the literature. These algorithms raise issues of implementation: the data structures that represent chromosomes and the population as well as efficient code for the algorithm's operations.

The Schema Theorem [Holland 1992] provides a theoretical foundation for genetic algorithms. It indicates that patterns within chromosomes that contribute to above-average fitness tend to increase in frequency within the population.

The course then explores variations of the genetic algorithm. For example, chromosomes need not be strings of binary symbols; they may be strings chosen from some larger alphabet, permutations of symbols, or two-dimensional structures such as matrices. Chromosomes may be selected to be parents in other ways than fitness-proportional selection. For example, parents may be chosen according to *linearly normalized* probabilities that are proportional to each chromosome's *rank* in the population or in *tournaments* of randomly selected chromosomes in which the most fit becomes a parent. Chromosomes may be combined by a variety of crossover operators, and randomly modified by a variety of mutation operators. A generational GA repeatedly builds a new generation of offspring, which replaces the current population; a *steady-state* GA inserts each offspring chromosome into the population immediately, deleting an existing chromosome to make room for the newcomer.

Another important issue is a GA's parameters, such as its population size and the probabilities with which its operators are applied. Researchers have described several strategies for setting these values. Having described these design alternatives, the course then examines GAs for several problems, such as the traveling salesman problem [Reinelt 1994].

Biological principles and processes have inspired other computational approaches to problem-solving, which are now grouped with genetic algorithms under the title *evolutionary computation*. *Evolution Strategies* (ES) [Bäck 1996, Schwefel 1995] addresses problems of numerical optimization using populations whose chromosomes are strings of floating-point values. *Evolutionary Programming* (EP) [Bäck 1996, Fogel 1995] also processes strings of floating-point values. Its only operators are selection and mutation; EP does not include crossover. *Genetic Programming* (GP) [Koza 1992] extends genetic algorithms to search a space of *programs* for one that solves a problem in general. A chromosome in GP is a tree that represents a program, a collection of such trees forms the population, and crossover and mutation operators are described for them.

The course concludes with students' presentations of research papers. [Figure 3] summarizes the course syllabus.

- Problems of search and optimization.
- Overview of evolutionary biology and genetics.
- Selection, crossover, and mutation.
- A generational GA for a combinatorial problem.
- Implementing a genetic algorithm.
- A generational GA for function optimization.
- Theory: the Schema Theorem.
- Variations on the theme.
- Examples of applications.
- Related paradigms.
- Presentations of research papers.

Figure 3: The syllabus for a senior/graduate course on genetic algorithms.

3.1 Assignments

Throughout the course, students are required to read papers in the literature and write short summaries of them.

In addition, there are two projects. In the first, each student writes and documents a binary-coded, generational GA for a combinatorial problem like PARTITION. The assignment includes comparing the GA's performance with that of a greedy algorithm for the chosen problem and writing a paper, in the style of a conference paper, describing the GA, the greedy algorithm, the comparison, and conclusions drawn from the investigation.

The second project consists of two parts. In the first part, each student writes a proposal describing a research project based on his/her reading and relating it to the literature. In the second part, the student carries out the project and writes it up, again in the style of a conference paper. A call for papers, reviewing forms, and instructions to authors are modeled on materials from conferences, and each student reviews the papers of two or three others. Students present their papers in conference-like sessions during the last week of the term, and the papers are distributed in a proceedings booklet.

The course includes traditional exams: two one-hour exams during the course and a comprehensive final exam at its conclusion.

3.2 Experience

The course has been offered twice and is running now (April, 1998). Students have been enthusiastic about it, and their projects have often shown both imagination and diligence. Titles of their papers have included "Classification by Linear Discriminant Analysis Using Genetic Algorithms," "Analyzing Operator Performance in the Traveling Salesman Problem," "DNA Restriction Fragment Map Reassembly Using a Genetic Algorithm," and "A Genetic Algorithm for the Frequency Assignment Problem."

4 Resources

In spite of the relative youth of genetic algorithms as a well-defined area of investigation, many texts, journals, and other materials on GAs are available. A valuable resource for anyone preparing a course on GAs is the book of contributed course materials assembled by Koza [Koza 1995]. This book includes syllabuses and other materials from about 30 university courses on GAs. The following subsections describe some other materials.

4.1 Texts

Goldberg was the first author to offer a text on genetic algorithms, and it has become a standard reference [Goldberg 1989]. However, the field has developed since its appearance, and an instructor using it as a text should supplement it with more recent material. This book was used in the first instance of CSCI 475/575. Davis' text [Davis 1991] is divided into two parts. The first part introduces genetic algorithms by describing the incremental development of a steady-state GA for a standard problem in function optimization. The second part consists of contributed papers describing applications of GAs. Mitchell has written an introductory text on GAs [Mitchell 1996]. It is recent and current, though its organization does not match that of the course described here. It was used in the second instance of the course.

Another text on GAs is by Michalewicz [Michalewicz 1996]. This book is well organized for a course and addresses all the main threads of GA research. It is currently in its third edition, and so is up-to-date, and includes a feature of particular interest to students: an appendix suggesting a variety of research project ideas. This text was selected for the current instance of CSCI 475/575.

A text that addresses the three styles of evolutionary computing—GAs, EP, and ES—with equal weight is [Bäck 1996], which also explores the underlying theories of the three approaches. Finally, the first 70 pages of Koza's book on genetic programming [Koza 1992] constitute an excellent introduction to genetic algorithms.

4.2 Journals

The first journal devoted to genetic algorithms and related topics was *Evolutionary Computation*, published quarterly by MIT Press Journals and now in its fifth year. Volume 1, Number 1 of *IEEE Transactions on Evolutionary Computation*, also quarterly, appeared in April of 1997, and a call for papers has been issued for a journal to be be published on the Internet called *Evolutionary Optimization*.

Adaptive Behavior, also from MIT Press Journals, addresses adaptation in both natural and artificial systems. Articles describing genetic algorithms for various problem domains can be found in journals devoted to those domains, such as network design, neural networks, operations research, the mathematics of finance, and optimization.

4.3 Proceedings

The first conference devoted to genetic algorithms was the First International Conference on Genetic Algorithms (ICGA) in 1985. Further ICGAs have occurred in subsequent odd-numbered years, and their proceedings, published first by Lawrence Erlbaum Associates and then by Morgan Kaufmann Publishers, constitute both a record of the development of GA research and applications and a guide to the most recent work in the area. A corresponding conference that meets in Europe in even-numbered years is Parallel Problem Solving from Nature (PPSN). Springer-Verlag published the proceedings of the first, third, and fourth PPSNs, Elsevier the second.

Other conferences devoted to some aspect of evolutionary computing include Evolutionary Programming, Genetic Programming, Foundations of Genetic Algorithms, and the IEEE International Conference on Evolutionary Computation. Papers and sessions on genetic algorithms and related topics can also be found at conferences dealing with artificial intelligence, optimization, and applications.

4.4 On-Line Digests

Among the on-line materials addressing genetic algorithms and related topics are several digests distributed via e-mail. GA-List addresses genetic algorithms and related topics and appears approximately weekly. To subscribe, send a message to `GA-List-Request@aic.nrl.navy.mil`. Similarly, EP-List is devoted to evolutionary programming; to subscribe, send a message to `EP-List-Request@magenta.me.fau.edu`. Genetic Programming discusses GP; to subscribe, send a message to `Genetic-Programming-Request@cs.stanford.edu`.

4.5 Software

Many genetic algorithm implementations are currently available, both free and commercial. Among the free options are LibGA from the University of Tulsa [Corcoran and Wainwright 1995], a Pascal implementation of the simple GA from Goldberg's text at the Illinois Genetic Algorithm Laboratory's web site (`http://gal4.ge.uiuc.edu/`), and a variety of GA programs in many languages at the GA-List web site (`http:// www.aic.nrl.navy.mil/galist/`). A search of the World Wide Web will identify many other sources as well, but I recommend against using such software in a course. Genetic algorithms are not difficult to write, and the experience of writing a GA is pedagogically valuable; one who can write such a program will understand it far better than one who can only use it.

5 Conclusion

This paper has briefly described genetic algorithms and a course for seniors and graduate students describing GAs. The course introduces the idea of population-based evolutionary search as used in GAs, then describes GAs for some typical problems, implementation issues, theory, variations of GAs, applications, and the other paradigms. Students read conference and journal papers and carry out two projects, the second of which is a research project. Both require papers and the second an oral presentation as well. Now in its third iteration, the course has been well received, and students have done good work in it. Several texts are available for such a course, as well as journals, conference proceedings, and on-line resources.

6 References

[Bäck 1996] Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice.* New York: Oxford University Press.

[Corcoran and Wainwright 1995] Corcoran, A. L. & Wainwright, R. L. (1995). Using LibGA to Develop Genetic Algorithms for Solving Combinatorial Optimization Problems. *Practical Handbook of Genetic Algorithms: Applications* (L. Chambers, Ed.), volume I, Boca Raton, FL: CRC Press, 143–172.

[Davis 1991] Davis, L. (1991). *Handbook of Genetic Algorithms.* New York: Van Nostrand Reinhold.

[De Jong 1975] De Jong, K. (1975). *Analysis of the Behavior of a Class of Genetic Adaptive Systems*, PhD Thesis, University of Michigan, Ann Arbor, MI.

[Fogel 1995] Fogel, D. B. (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* New York: IEEE Press.

[Goldberg 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Learning.* Reading, MA: Addison-Wesley.

[Holland 1992] Holland, J. (1992). *Adaptation in Natural and Artificial Systems.* Cambridge, MA: The MIT Press.

[Koza 1992] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: The MIT Press.

[Koza 1995] Koza, J. R. (Ed.) (1995). *University Courses on Genetic Algorithms 1995.* Stanford, CA: Stanford Bookstore.

[Michalewicz 1996] Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*, third edition. Berlin: Springer-Verlag.

[Mitchell 1996] Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* Cambridge, MA: The MIT Press.

[Oliver, Smith, & Holland 1987] Oliver, I. M., Smith, D. J., & Holland, J. R. C. (1987). A Study of Permutation Crossover Operators on the Traveling Salesman Problem. *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (J. J. Greffenstette, Ed.). Hillsdale, NJ: Lawrence Erlbaum Associates, 224–230.

[Reinelt 1994] Reinelt, G. (1994). *The Traveling Salesman: Computational Solutions for TSP Applications.* Berlin: Springer-Verlag.

[Schwefel 1995] Schwefel, H.-P. (1995). *Evolution and Optimum Seeking.* New York: John Wiley & Sons.

Beyond a First Course in Windows Programming:
Why, How, and What?

James P. Kelsh
Department of Computer Science
Central Michigan University
Mt. Pleasant, MI 48859
James.Kelsh@cmich.edu

Roger Y. Lee
Department of Computer Science
Central Michigan University
Mt. Pleasant, MI 48859
lee@cps.cmich.edu

Our department has been teaching a one-semester course in Windows programming on a regular basis for three years, offering one section each spring semester. After extensive "Independent Study" projects with our graduate students, we are now introducing a second course. This paper explains why we need two courses, how we formed our plans for the second course, and what topics we consider important in it.

Beyond a First Course in Windows Programming: Why, How, and What?

James P. Kelsh
Department of Computer Science
Central Michigan University
Mt. Pleasant, MI 48859
James.Kelsh@cmich.edu

Roger Y. Lee
Department of Computer Science
Central Michigan University
Mt. Pleasant, MI 48859
lee@cps.cmich.edu

Introduction

Our department began teaching a one-semester course in Windows programming on a regular basis in 1994, offering one section of the course each spring semester. For three years, it was a “Special Topics” course, only now being offered with its own name and number. Yet we already feel the need for a second course. This paper explains why we need two courses, how we formed our plans for the second course, and what topics we consider important in it.

Why a Second Course?

Visual C++ imposes a steep learning curve. When we teach Windows Programming with Visual C++ at the undergraduate level, our students can’t move beyond relatively straightforward programs in one semester. They can respond to events (mouse clicks, menu selections, etc.), and display screen and printed output, but time does not permit reaching, for example, database techniques. Ironically, while we program in a graphical user interface, we can barely touch on computer-generated graphics in a first course. This seems insurmountable if we continue to believe that students should be allowed to learn Windows programming without taking a course in database or graphics as a prerequisite. This raises a new question: Why stick with Visual C++?

Why Not Visual Basic?

Many observers point out that Visual Basic introduces people to Windows concepts much faster than Visual C++ does. We have previously moved one of our service courses from QBasic to Visual Basic, and instructors who are familiar with Windows programming in Visual C++ are amazed to see how quickly they can introduce Windows concepts to less skilled programmers when the medium is Visual Basic. Our program to prepare secondary teachers embraces Visual Basic with the hope that our graduates will endear themselves to their school districts for being able to quickly develop attractive and easy-to-learn Windows programs with Visual Basic. Even the purists admit that much of the stigma of “Basic” is gone now that Visual Basic v5.0 can compile to native code with alarming improvements in efficiency.

Still, Visual C++ remains the first platform for new Windows technologies. Maybe this is why our industry contacts continue to seek Visual C++ experience when they hire our graduates. We would like to acquaint our students with more development tools. But with limited time in the curriculum, Visual C++ seems more valuable to our students. An added benefit for the more adventurous is the opportunity to read Microsoft's source code and study the implementation of the Foundation Classes. (This is not possible with Visual Basic.)

How to Design the Course?

After teaching Windows Programming three times, we started noticing two problems. First, having a new version of the compiler and of the class library each year meant that the course was a continually moving target. Every concept had to be tested again to see how much of it still worked the way it did a year ago. Partially because of that, and partially because our other courses were merely TTY programming (not reinforcing Windows techniques), we realized that by the end of the semester we had told our students very nearly everything we knew about Windows programming. Certainly there was more to cover, but the one course was not exposing us to it.

During the summer 1997 term, we experienced the bonus (and bane) of our institution's status (mostly undergraduate, but supporting an M.S. program): Our graduate students, realizing the value of Windows experience, enrolled in Independent Study projects in higher than usual numbers.

Independent Study / Special Topics
"Special Topics" courses are too new or too transient to be part of our permanent course offerings. These are taught as part of the instructor's regular teaching load. These courses are frequently used to introduce and test new courses for the curriculum. Independent Study students, on the other hand, are rarely counted toward teaching load. Many of these are familiar projects for the instructor; with Windows programming, few Independent Study projects are familiar. Thus, with Windows, Independent Study is a way to get an idea for a course, and Special Topics courses are ways to test the idea. In the past, we have usually had about 5 graduate students doing Independent Study with each section of our Windows Programming course. In summer 1997 we had an unusually large number, almost enough to populate a course, studying Windows programming. They represented a mix of beginners and repeating Windows programmers. Although this Independent Study was uncompensated (financially), the enrollment provided the impetus for a summer seminar to study:

1. The left-over chapters in our textbooks.
2. The topics mentioned by our guest speakers.
3. Our own interests.

What Topics to Include?

The topics our seminar drifted into included:

1. Database: Most of our graduate students had taken our (500-level) database course and realized that a large portion of application programming involves database applications. MFC provides classes (CRecordset, CDaoRecordset) to provide an almost dBase-like flavor, with SQL lying underneath. But we can't include this in our introductory course without imposing a database prerequisite.

2. COM: Microsoft's Component Object Model is the basis of their modern technologies for dynamic (run-time) linking and OLE (Object Linking and Embedding). While dynamic linking of specific files has been available for years, COM provides the facility for a program to ask the run-time environment what software tools are available to perform certain tasks. But the current

implementation is beyond our average beginner. (Reports suggest that Windows 98, with COM+, may be somewhat easier to introduce. The moving target continues to move ....)

3. Animation: Some of us "old-timers" are surprised to see that even students who have not studied graphics are interested in (and soon become accomplished at) animation. Knowledge of computer graphics techniques for rendering an image are not required if the programmer has a two-dimensional image and needs to call an application programming interface to move it smoothly across the screen. Even "three-dimensional" graphics (we can lament the fact that the term "three-dimensional" now refers to perspective and the ability to rotate objects rather than to stereoscopic views and binocular vision, but we seem unable to change that fact) are manipulated by calls to a graphics engine. (We have one undergraduate student currently involved in an Independent Study project with Microsoft's Direct 3D graphics engine.)

The seminar was less successful than hoped, and several students' work remained incomplete. These incomplete grades are only now being removed. Still, the experience helped distill the ideas for a second course in Windows Programming.

The Course-To-Be (Fall 1998):

We are now offering a 500-level ("Special Topics") course scheduled for Fall 1998. The 500-level is the "twilight zone" common to many institutions with mixed student bodies: Courses which may be taken by undergraduates and graduates. Our institution states that courses numbered from 300 to 599 are officially undergraduate courses that may be taken for "graduate credit," but insists that graduate students must face higher grading standards.

The prerequisites for the course are our (500-level) database course and either our undergraduate course in Windows programming or graduate standing.

The Mixed Audience

Some of our undergraduate students may take this as a second course in Windows programming, while some graduate students may take it as a first course. Thus, we will begin with basic Windows "event-handling" programs. The units shown below will probably take about one week each:

1. Respond to left and right button mouse clicks with a "Message Box" telling how many times the respective button has been clicked.
2. Design a "dialog:" Respond to a left-button click by displaying the dialog, gathering information from the dialog, and adding the information to a list. (A separate function [OnDraw"] ensures that the current state of the list is always displayed in the application's window.)
3. Scrolling and Printing: CScrollView does much of the work for scrolling windows. Printing and pagination challenge most of our undergraduates. We hope our graduate students, with better mathematical backgrounds, will absorb it more quickly.
4. Menus and Help: Handlers for menu commands are routine after handling Windows messages. Help is currently a significant task involving RTF (rich text format) files and a separate "help compiler," but rumors suggest that HTML may soon become the *de facto* standard for Windows help files. Context-sensitive menus, summoned by *right*-clicking, are the currently fashionable way to obtain a list of actions appropriate to an object.

Readers may note that this omits the "serialization" ("serially" reading data from disk or storing data to disk) focus which many consider important. It does not introduce CObList, CArray, or other useful container classes. While not disputing the importance of such concepts in an introductory Windows class, we note that concentrating on databases removes the necessity for serialization (unless a user chooses to attach a set of preferences to the tables of a database), and the database is its own container. Even this "lean and mean" overview must be presented at a quick pace for graduate students, while providing a review for undergraduates. We hope to call the students' attention to several other classes they would be wise to study.

Proceeding to databases:

5. Review of database concepts: Third normal form, SQL syntax, and normalization. An exercise will include reducing a flat-file database to third normal form.
6. ODBC. Flat-file recordsets. The CRecordset class maintains a list a records resulting from a query. The SQL programmer may think of it as a cursor. DBase programmers will feel comfortable with MoveFirst, MoveNext, and IsEOF functions, leading to code like the following to display a list of authors:

   ```
   name_set.Requery();
   name_set.MoveFirst();
   y = START_Y;
   while (!name_set.IsEOF() )
   {
    x = START_X;
    pDC->TextOut(x, y, name_set.tell_Lname() + ", ");
    x = MARGIN2;
    pDC->TextOut(x, y, name_set.tell_Fname() + " " +name_set.tell_Mname() );
    y += ROW_HEIGHT;
   }
   ```

   CRecordView (akin to a Visual Basic "form") vs. using a recordset for data and a traditional CView for display. Microsoft presents CRecordView as a Visual C++ programming technique that resembles using a "data control" in Visual Basic. For greater control of normalization and query techniques, we favor "TextOut" calls in a traditional "view" class's OnDraw function.

7. Relational databases. One recordset for several tables vs. separate recordsets and programmer-implemented joins. Query parameters and optimization.
   If a database "view" (without update) is needed, one recordset can represent the join of several tables. However, to update individual tables requires separate recordsets, with join accomplished by programmer code. This should reinforce the students' understanding of the natural join. It also demonstrates graphically the performance penalty resulting from changing a query (Microsoft closes and reopens the recordset) rather than using query parameters.
8. Relational properties and joins, continued.
9. ODBC vs. DAO. This is customarily presented as a performance question. The programming is virtually unaffected by the choice of ODBC or DAO.

Component Object Model:

Microsoft's Component Object Model. Review of separately compiled classes. Dynamically linked objects (Microsoft DLLs). Using a "normalization" object (prepared by an undergraduate student currently involved with an "Independent Study" project).

Component objects can be viewed as a generalization of separate compilation: After we feel comfortable writing code in separate files and linking the object code together, we can consider leaving the code unlinked. At run-time, some part will feel the need for another part. If the "other part" was prepared as a dynamic link library, it can be linked on demand at run time. But the component object model moves beyond this to allow a program to determine at run-time which libraries it needs, even extending to libraries not written when the program was written!

After the preliminaries of creating a "class factory" and generating an object, the program obtains an "interface." The interface (like a menu in a restaurant) identifies a list of functions that can be called and allows access to other interfaces stored in the object. (Think of a dinner menu declaring that a "breakfast" menu is available.) From the starting interface I(Unknown), the program can ask about any other interfaces available. An object which contains another object can announce the interfaces of the contained object.

But this alone would not let the program access objects that did not exist when the program was written. From the Open Software Foundation's "Universally Unique" IDentifiers to Microsoft's "Globally Unique IDentifiers, objects can assign themselves identifying numbers. Moreover, the Windows Registry can maintain categories and let objects register themselves in terms of the services (interfaces) they can provide. Thus, a program can ask the Registry for any component object able to provide specified services.

This will probably occupy the rest of the semester, but we will introduce animation if time permits. Most of the projects will be in a database context.

Course Materials:

No text known to us seems to cover this mix of topics. We may prepare our own supplements, referring the students to various texts on reserve in the library to cover the diverse aspects. [Kruglinski, 1997] has been an adequate introduction to Visual C++ programming, and includes an introduction to COM. However, database coverage is minimal. [Silberschatz, et al., 1997] is a more than adequate treatment of database principles. [Brockschmidt, 1995] is an encyclopedic treatment of COM, while [Rogerson, 1997] is much more easily readable. [Thompson, 1996] puts several layers on top of Windows graphics programming. This may help the student get something running sooner, but may be frustrating if the student cannot get far enough in the book. But the combined list prices of these books exceeds $250, and we seldom ask even our graduate students to spend that much on books for one course!

References:

[Brockschmidt, 1995] Brockschmidt, Kraig. (1995) Inside OLE, Second Edition. Redmond, Washington: Microsoft Press.

[Kruglinski, 1997] Kruglinski, David. (1997) Inside Visual C++. Redmond, Washington: Microsoft Press.

[Rogerson, 1997] Rogerson, Dale. (1997) Inside COM. Redmond, Washington: Microsoft Press.

[Silberschatz, et al., 1997] Silberschatz, Abraham, Korth, Henry F., & Sudarshan, S. (1997) Database System Concepts, Third Edition. New York, NY: McGraw-Hill.

[Thompson, 1996] Thompson, Nigel. (1996) 3D Graphics programming for Windows 95. Redmond, Washington: Microsoft Press.

Time and Space Independent Learning
On-line Course Offerings at a Small Liberal Arts College
A Status Report

Michael G. Konemann
Computer Science Department
Carroll College
mgk@carroll1.cc.edu

Lynda A. Thomas
Computer Science Department
Carroll College
lthomas@carroll1.cc.edu

Gerald L. Isaacs
Computer Science Department
Carroll College
gli@carroll1.cc.edu

Abstract

At Carroll College, approximately 60% of the class sections of the Computer Science department are now taught at night and on weekends. We are increasingly finding that to best serve our students, we need to offer courses that fit in with their schedules. Over the last two years, this need and the ease with which courses can be offered on the web, have led us to offer some sections of some of our courses through the medium of the World Wide Web. In this paper, we discuss our experiences with these web offerings. We envision this presentation as an opportunity to share our experiences at a small liberal arts college with others in similar situations, and to gain from their experiences in turn.

Time and Space Independent Learning
On-line Course Offerings at a Small Liberal Arts College
A Status Report

Michael G. Konemann
Computer Science Department
Carroll College
mgk@carroll1.cc.edu

Lynda A. Thomas
Computer Science Department
Carroll College
lthomas@carroll1.cc.edu

Gerald L. Isaacs
Computer Science Department
Carroll College
gli@carroll1.cc.edu

Introduction

At Carroll College, approximately 60% of the class sections of the Computer Science department are now taught at night and on weekends. We are increasingly finding that to best serve our students, we need to offer courses that fit in with their schedules. Over the last two years, this need and the ease with which courses can be offered on the web, have led us to offer some sections of some of our courses through the medium of the World Wide Web. In this paper, we discuss our experiences with these web offerings.

Carroll College is located in Waukesha, Wisconsin, which is a mid-size city (approximately 57,000 residents) located 20 miles west of Milwaukee in the Greater Milwaukee Metropolitan Area. We have a large non-traditional part time student population, as well as approximately 37% commuting students. This group of students has a different set of needs than the traditional on-campus student.

We do not regard on-line learning as a replacement for more ëtraditionalí instruction, but rather as a supplement and/or an alternative. As others have pointed out: ìThe use of hypermedia is associated with the idea of learner-control -- the learners decide which topics they will view, in what order they will view the topics, how the topics are related, how long they will spend on each topic, etc.î [Sweany et al. 1996]. Learner-control provides an advantage, which may be countered, however, by a feeling of disorientation or ëinformation overload.í How these two forces have played out in our classes is the topic of this paper.

Larger schools with more resources are increasingly offering help to faculty who wish to build web courses (see for instance [1], [2] and [3]). We envision this presentation as an opportunity to share our experiences at a small liberal arts college with others in similar situations, and to gain from their experiences in turn.

Carroll College in a Nutshell

Carroll College is a private, non-profit institution, offering programs leading to the bachelor's and master's degrees. The Carnegie Foundation for Education classifies it as a Baccalaureate College II. The campus has more than 1,700 full-time students, as well as more than 900 part-time students. Additionally, there are 80-100 graduate students on the Carroll campus. Carroll College realizes that personalized education is the special province of a small college and recognizes the variety of students' individual needs and preferences.

Why We Started Web Based Courses at Carroll College

One of the degree-specific requirements at Carroll College is that anyone pursuing a Bachelor of Science degree must demonstrate competency in the usage of Information Technology. One way of demonstrating the required competence is to earn credit in Computer Science 105 and Computer Science 107. The titles and descriptions of these courses are as follows:

CSC105: *Introduction to Information Technology and The Internet* (2 credits)

Prerequisite for all computer science courses at Carroll. It provides students with an introduction to the three environments at Carroll: UNIX, PC and compatibles, and Macintosh. Special emphasis will be placed on computer ethics, electronic mail, WiscNet and the Internet. Word processing and spreadsheet applications will be presented in a hands-on learning environment.

CSC107: *Problem Solving using Information Technology* (2 credits -- Prerequisite CSC105)

This course provides a foundation in problem solving skills using Information Technology. Applications covered include spreadsheet functions and macros, PowerPoint, and Internet Programming.

Prior to introducing the opportunity to take "web-based" sections of these classes, students would take these courses in a more traditional setting. All sections of the courses met in classrooms equipped with 20-25 machines (either Windows-based or Mac-based). All classrooms have an instructor's machine, which can be projected. Some of the sections of the course are taught by full-time faculty members, while others are taught by adjuncts. In all cases, some of the class time consists of students learning applications and skills as the instructor demonstrates.

Several years ago the Computer Science Department moved away from the traditional didactic curricular approach, realizing that technology is best learned by experiential learning. Nearly all classes in our curriculum, and CSC105/CSC107 in particular, adhere to the following [Wagschal 1997]:

instructors guide, motivate, facilitate	*rather than*	instructors lecture while students listen
topics are integrated	*rather than*	content is pigeon-holed
curriculum is problem-centered	*rather than*	content is fact-centered
many rich sources for learning, accessible electronically using a variety of media	*rather than*	teachers are the primary source of knowledge
student success is presumed when students solve problems, communicate ideas, present information, learn how to learn	*rather than*	student success is presumed when students remember and can report back
learning is everybody's business and takes		schools are insular, largely

place everywhere; computers connect the world to the learner and the learner to the world.	*rather than*	separated from the rest of the community

All of these beliefs guide our instruction in our ìin-classî sections of 105/107, and they made web-based sections of these classes attractive.

We encounter two problems with the "in-class" sections of the course:

ï Instructors need to teach a classroom full of students with wide, varying degrees of knowledge and experience, and the diversity seems to continually increase. So, no matter what pace the instructor sets, there are some students who are frustrated by the slow pace, and there are some students who are frustrated because they can't keep up. Both types of students are good candidates for taking the course independently on the web.

ï With 80-90% of Carroll students pursuing Bachelor of Science degrees (therefore requiring these courses), and a large percentage of Carroll students being non-traditional students, there are many students who have scheduling conflicts trying to take these required courses.

Those were the primary motivations for creating web-based sections of CSC105 and CSC107. In addition, we currently offer web-based sections of CSC106, a course designed to introduce students to graphic communication technology, and programming courses in Visual Basic and Visual C++.

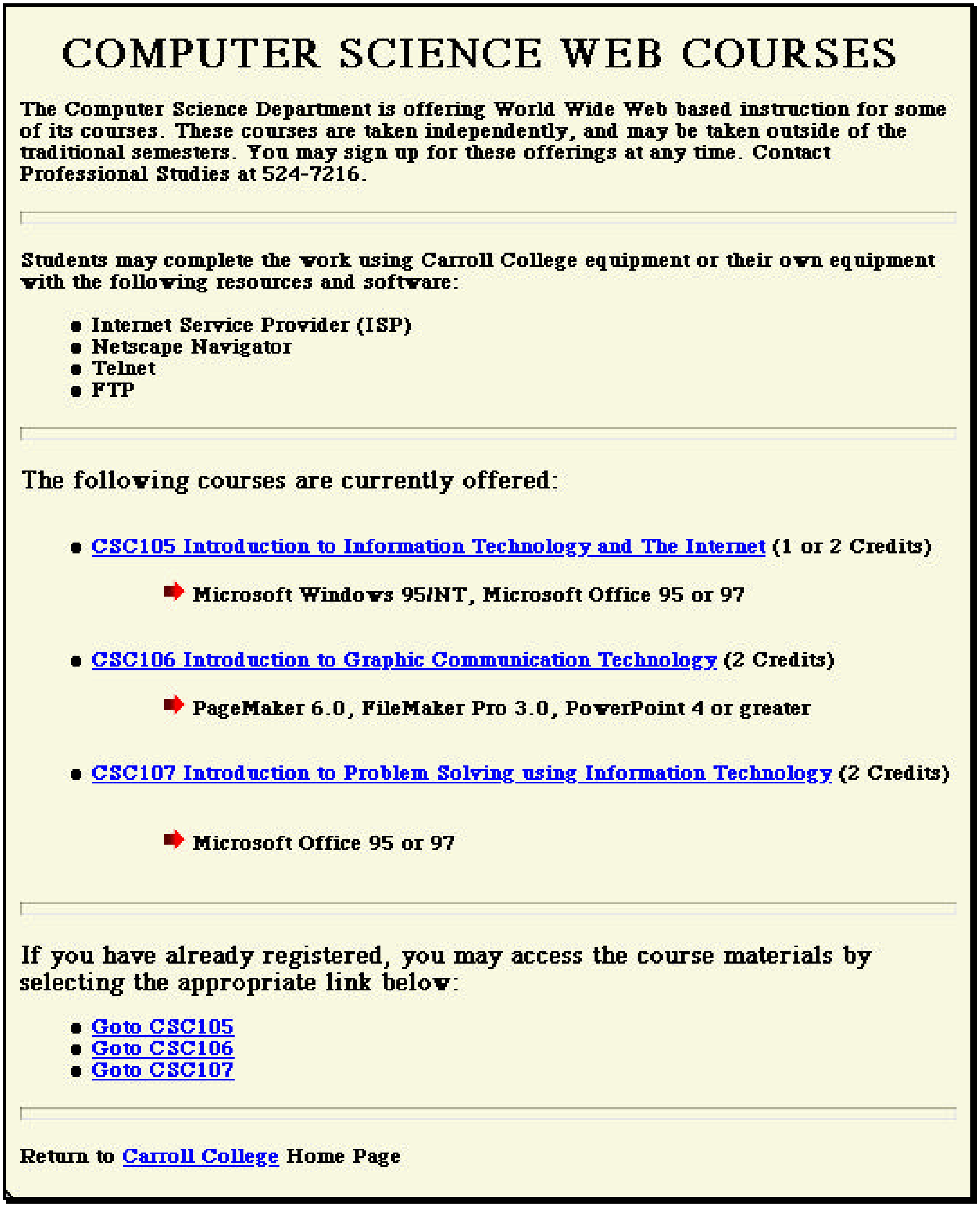
Who can take Web Classes?

Beginning in 1998, students must apply to take these independent web courses (see Appendix A). During the first year of these courses, it was noted that a number of students did not complete all of the assignments. Upon consultation with the registrar, it was noted that some of the students had a track record of being very poor students. It was felt that they had virtually no possibility of completing this work. So now every student must first fill out an application (see Appendix A) providing evidence of their ability to work independently.

Descriptions of Web Sections of CSC105, CSC106, and CSC107

Each of these classes is a two-credit class, which traditionally meets four hours a week for seven weeks. (However, CSC105 may be taken in one-credit pieces.) It was felt that students should be given two months to complete the requirements for each credit of work. Students who have not completed the course within the last two weeks of the period are notified by the registrar that they are in danger of failing. The student then must either drop the course or indicate how much longer they will need. This late drop date is being allowed because it is currently felt that these World Wide Web sections are experimental. It is expected that a more traditional drop date will be adopted in the future.

Students interested in taking web-based sections of courses at Carroll College are encouraged to visit the following web page:



COMPUTER SCIENCE WEB COURSES

The Computer Science Department is offering World Wide Web based instruction for some of its courses. These courses are taken independently, and may be taken outside of the traditional semesters. You may sign up for these offerings at any time. Contact Professional Studies at 524-7216.

Students may complete the work using Carroll College equipment or their own equipment with the following resources and software:

- Internet Service Provider (ISP)
- Netscape Navigator
- Telnet
- FTP

The following courses are currently offered:

- CSC105 Introduction to Information Technology and The Internet (1 or 2 Credits)
  - ➡ Microsoft Windows 95/NT, Microsoft Office 95 or 97
- CSC106 Introduction to Graphic Communication Technology (2 Credits)
  - ➡ PageMaker 6.0, FileMaker Pro 3.0, PowerPoint 4 or greater
- CSC107 Introduction to Problem Solving using Information Technology (2 Credits)
  - ➡ Microsoft Office 95 or 97

If you have already registered, you may access the course materials by selecting the appropriate link below:

- Goto CSC105
- Goto CSC106
- Goto CSC107

Return to Carroll College Home Page

If the student follows the *CSC105 Introduction to Information Technology and The Internet* link (or equivalent links), then the student is presented with a complete syllabus for the class, including course description, detailed objectives, grading criteria, and textbooks. If the student follows the *Goto CSC105* link (or equivalent links), then the student is presented with the following information:

This section of CSC105 is for those students who wish to work in an independent manner. Information and assignments will be provided via The Internet. Students will be required to e-mail assignments. Questions, answers, and information will be provided electronically, via phone, or by appointment.

Work may be completed using Carroll College's computer labs or on the student's computer provided they have the following: Microsoft Windows 95/NT, Microsoft Office 95/97, an Internet Connection (ISP), World Wide Web Browser.

CSC105 may be taken for either one or two credits. CSC105-1 and CSC105-2 each consist of 1 credit of work. Students will have up to two months per credit to complete the assignments and a final exam to be given at Carroll. The final exam will be on-line and mastery based. Those students taking both credits will only have one exam. Grading will be based upon completion of assignments (70%) and the final exam (30%).

The First Credit of CSC105 (or CSC105-1) consists of:

ï Introductory meeting to ensure understanding of e-mail and an Internet browser
ï Fundamental Microsoft Windows 95/NT Concepts
ï The Internet and World Wide Web Browsing
ï Library and Data Base Repositories
ï Microsoft Word Fundamentals

The Second Credit of CSC105 (or CSC105-2) consists of:

ï Microsoft Excel Fundamentals
ï Advanced Research Skills using Internet Tools
ï Advanced Microsoft Word
ï Advanced Microsoft Windows 95/NT Concepts

Contact Dr. Gerald L. Isaacs for details at 524-7148 or via e-mail at gli@carroll1.cc.edu.

To sign up for this offering, call Professional Studies at 524-7216.

GOTO Course Materials

Following the *GOTO Course Materials* link takes students to the specific assignments for the course. Appendix B provides a sample assignment. These pages are only accessible to students who have registered for the course. Since the materials are housed on a Windows NT 4.0 server, it was relatively easy to create groups for each class and then give access to the materials only to that group. Class lists are provided by Information Technology Services once a month. The World Wide Web sections begin at the beginning of each month with an in class introductory session. This session tries to ensure that all students understand the ground rules and where they can obtain assistance virtually any time. However, the help may not be immediate in the case of e-mail requests.

Summary of Hardware and Software Requirements

From the student's perspective, all that is required is what is listed above. That is, they need access to a computer with the following:

ï an Internet Connection (ISP)
ï Microsoft Windows 95/NT (or a Macintosh, if they so choose)
ï Microsoft Office 95/97
ï World Wide Web Browser (Netscape or Microsoft Internet Explorer)
ï other applications specific to the particular course

Students are welcome to use Carroll's machines for part or all of their work. Most students end up using their own machines for the majority of their work.

From the perspective of faculty members offering web-based courses, the hardware/software requirements are basically the same. In addition to the above, the instructor must have control to make course materials available to the students who have formally registered for the course.

Profiles of students participating in these courses

Having the courses on-line allows the students to take the courses at their convenience, rather than being constrained to a traditional semester. As of this writing, 80 students have enrolled in web-based sections of various courses. There are several reasons why the students are taking the web-based sections. Many of the students are non-traditional adult students (approximately 70%), facing typical obstacles that constrain the time they have available to take courses. Occupational demands that include extensive travel and scheduling conflicts, as well as family needs are typical reasons why students choose the web-based courses. Some students take the web-based section because they prefer to work at a faster (or slower) pace than the non-web-based sections allow. Some students take the web-based sections because no other sections fit their schedule. Web-based sections have also been taken by members of the United States Olympic Speedskating Team. Since their training facility is nearby, Carroll has entered a partnership with members of the speedskating team, allowing them to schedule classes around their busy training schedule.

Preventing cognitive overload

While the learner-controlled nature of the web-based courses are a big advantage, the disadvantage of having the student face cognitive overload needs to be addressed. As stated in [Sweany et al. 1996], ìThe learner will have trouble comprehending brand-new information if he or she is having to adapt to an unfamiliar vehicle for presentation of the information.î We combat that in our web-based courses in the following ways:

1. All students are required to attend a meeting with the lead instructor before starting the course material. At this time it is verified that the student understands how to use their e-mail account (which has been created and tested before this first meeting), how to contact the instructor with questions, how to submit assignments, and basically gives the student a comfort zone before starting.

2. A handout is provided which stresses the independent nature of the experience, expectations of the class, and various methods where help can be obtained (see Appendix C). For example, there are tutors available for students to either call on the telephone, or sit down with for one-on-one assistance.

3. One of the textbooks for the courses is an in-house publication entitled ìTechnology for the 21st Century: A Carroll College Guide.î This guide, written by the Computer Science Department faculty, covers material specific to the usage of information technology tools at Carroll College. One of its goals is to keep novice users from getting disoriented.

4. Most of the textbooks we choose for the web-based courses include tutorials and exercises that guide the students in their learning. We make all of the necessary files available via FTP, describing the process to the students during their meeting described above. In addition, students can borrow a CD containing all of the necessary files and non-proprietary applications for the course, which they can copy onto their home computer.

5. Students receive timely responses to their questions and feedback from their submitted assignments. All questions submitted via e-mail are responded to within 24 hours. When assignments are submitted via e-mail, scores are posted within 24 hours. The gradesheet is available electronically from the course web page (with scores posted via anonymous ID numbers).

6. Students create virtual cohort groups as a means of exchanging ideas, information, and questions. Listservs are in place, which the students subscribe to as one of their first assignments.

Evaluation of students

Students taking web-based sections of Computer Science courses at Carroll College are required to take a final exam. The student must come to the Carroll campus, at their convenience, to ensure that the exam is taken in a proctored environment. The exam requires that the student demonstrate the level of their competencies on-line. The exam is basically the same exam taken by the students who take regular, non-web-based sections of the course. Although we have not done any statistical comparisons, we have clearly seen that the web-based studentsí performance and achievement equals or exceeds that of the non-web-based students. While the web-based students so far have done extremely well on the exam, there is a great deal of variance in how the regular classroom students perform.

What is the Future of Web-based instruction at Carroll College?

Our current web courses are completely asynchronous learning. There are no time constraints on when assignments are due. However, we have seen that this works as a disadvantage to some students who are not disciplined. In the future, we may offer some sections where students would be expected to complete each assignment within certain time constraints.

We expect student enrollments to increase in the existing web-based courses at Carroll College. The learner-controlled environment has been successful for the students who have gone through it so far. From a pragmatic point of view, if more students take the courses on the web, then fewer sections of the courses need be taught by adjuncts. With everything else being equal, fewer adjunct sections is a positive from the administration's point of view.

We find that most of the courses in our Computer Science curriculum could ultimately be web-based. We will be adding web-based sections of other Computer Science courses during the next term. Coming on-line are sections of our Visual Basic, Visual C++, and introductory web creation courses. It is expected that some of our adjuncts may even help in the creation and delivery of web courses. These individuals are currently working in industry and have the desire, expertise, and knowledge to create enjoyable and pertinent web classes.

As long as simultaneous communication and interaction between members of the class is *not required*, then the course has potential to be web-based in the simple way that our classes have been organized. That is, as long as the instructor can work asynchronously with the students, then a web-based approach should be successful. If student collaboration is required synchronously with the instructor or other students, then our simple solution would not be appropriate. Our current approach is not that of an instructor leading a Virtual Classroom, but rather more of a one-on-one relationship with the instructor mentoring individual students. However, we will experiment with virtual synchronous discussions held at different time periods throughout the experience. Some of these will require alternate times so that all can participate.

References

[1] Texas Tech University: http://www.tltc.ttu.edu/webbased/index.htm

[2] University of Illinois at Chicago: http://www.uic.edu:80/depts/adn/webclass/authoring/authoring1.html

[3] University of Minnesota: http://www.mes.umn.edu/~hoefer/coursdev/webasedi.htm

[Sweany et al. 1996] Sweany, Noelle D., McManus, Thomas F., Williams, Doug C., & Tothero, Kenneth D. (1996). *The Use of Cognitive and Metacognitive Strategies in a Hypermedia Environment*. Presented at EdMedia, Boston, MA.

[Wagschal 1997] Wagschal, Peter. (1997). *A University Without Walls in the Information Age*. Presented at The College Board, Phoenix, AZ.

Appendix A: Application for Independent Learning Experience

Students requesting permission to enroll in an Independent Learning Experience course, offered through the School of Professional Studies, must submit this completed form with course registration for review prior to beginning work associated with a specific class.

Name: ____

Address: ____

____

e-mail: ____________ **Telephone:** ______________

Carroll Major: ____________ **Academic Advisor:** ______________

Course Number: ____________ **Term/Month:** ______________

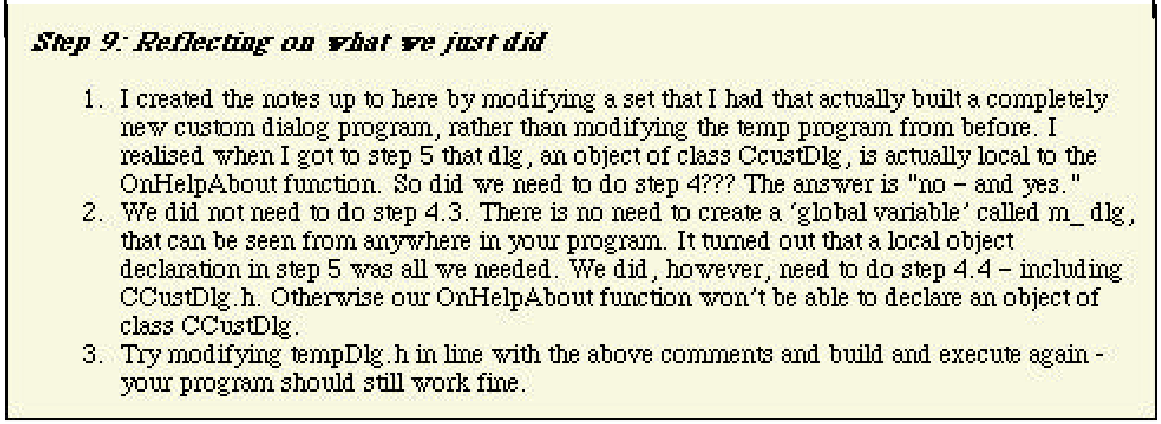
Use the space below to provide evidence of your ability to complete the work associated with a course where the primary course work is completed independently.

Signature: ____________________ **Date:** ____________________

Return to:
Registrarís Office
Carroll College
100 N. East Avenue
Waukesha, WI 53186

Appendix B: A Sample Assignment

●
●
●



Step 9: Reflecting on what we just did

1. I created the notes up to here by modifying a set that I had that actually built a completely new custom dialog program, rather than modifying the temp program from before. I realised when I got to step 5 that dlg, an object of class CcustDlg, is actually local to the OnHelpAbout function. So did we need to do step 4??? The answer is "no – and yes."
2. We did not need to do step 4.3. There is no need to create a 'global variable' called m_ dlg, that can be seen from anywhere in your program. It turned out that a local object declaration in step 5 was all we needed. We did, however, need to do step 4.4 – including CCustDlg.h. Otherwise our OnHelpAbout function won't be able to declare an object of class CCustDlg.
3. Try modifying tempDlg.h in line with the above comments and build and execute again - your program should still work fine.

Appendix C: Handout Describing Guidelines and Information

GUIDELINES & INFORMATION

Courses may be started at the beginning of each month. On-line cohort groups are created where students create discussions, digest problems, and examine solutions. Duration of the class varies depending upon the course and number of credits. Students enrolled in a two-credit course usually have up to four months from the beginning session to complete the material.

Students must have:

- A valid Carroll College Information Technology Services (ITS) computer account.
  Call 524-7229 if you do not have one. Every Carroll College student is eligible.
- Netscape (or another browser), Telnet, FTP installed on their computer.
- Internet Access via local or national Internet Service Provider (ISP).
- Books or other materials which are listed on each syllabi.
  Syllabi and objectives may be found at: http://www.cc.edu at the web classes page.
  The Carroll College Bookstore has books and may be reached at 414-524-7344.

On-line courses are not for everyone.

Students must be:

- Able to Work Independently.
- Motivated.
- Willing to ask for assistance when necessary.

Assignments and materials need to be accessed via the World Wide Web. Many information resources and repositories are available via the Carroll College Library's home page at http://www.cc.edu/library.

Help is available:

- Via e-mail with your Instructor.
- Via e-mail with advanced student tutors (tutors@carroll1.cc.edu).
- Via phone with your Instructor.
- Via phone 414-650-4969. See http://www.cc.edu/compsci for hours.
- At Carroll College in Main 14. See http://www.cc.edu/compsci for hours.

To register:

- Contact Professional Studies (414-524-7216).
- After you have registered and paid your fees, contact Dr. Gerald L. Isaacs to setup
  an initial meeting with your instructor (524-7148 or gli@carroll1.cc.edu).

Bringing Streaming Media to Campus

Aaron Korb
Doane College
Crete, NE 68333
akorb@doane.edu

Abstract

Bringing streaming media to campus was a project that brought RealAudio and RealVideo to Doane College. The installation of the media server was done as a senior project for a computer science major. This paper explains the process of how streaming media was proposed, installed, and the troubleshooting that was involved to make RealAudio and RealVideo possible for the college web site.

Bringing Streaming Media to Campus

Aaron Korb
Doane College
Crete, NE 68333
akorb@doane.edu

Abstract

Bringing streaming media to campus was a project that brought RealAudio and RealVideo to Doane College. The installation of the media server was done as a senior project for a computer science major. This paper explains the process of how streaming media was proposed, installed, and the troubleshooting that was involved to make RealAudio and RealVideo possible for the college web site.

Introduction

Bringing streaming media to campus can add new dimensions to a college's web site. This paper will describe the process used to enable Doane College to serve RealAudio and RealVideo across the web. The project, and this paper, can be outlined in three steps: the proposal, the installation, and the troubleshooting.

The Proposal

The proposal was the first step in the process. The research needed to make an accurate proposal took up the majority of the time. For this project, the idea of streaming media was originally intended for the campus radio station to allow broadcasting over the Internet.

The campus radio station had expressed interest in the idea after it was discussed in November of 1996 at the National Association of Collegiate Broadcasters convention in Providence, Rhode Island. This conference focused on the expansion of college radio into the age of the World Wide Web. The attraction to this is that a station that may normally broadcast 30 or 40 miles could broadcast around the world. The radio station and communications department seemed interested, but it was not known whether the campus network could handle the additional traffic.

Working on the project with the Doane Office of Technology in the spring of 1997 proved difficult. For undisclosed reasons, the director was against the project. Therefore, the process of gathering information was impossible. The problem continued until the end of the academic year. In June of 1997, a new director was hired and the project again became a possibility.

The campus network had undergone a major overhaul the year before, and with the installation of fiber optics and Ethernet to all buildings on campus, including the building where any RealAudio and RealVideo server would be held, the project seemed feasible. The campus is connected to the Internet with a single T1 line. In studying the amount of network traffic, the T1 had plenty of bandwidth to support the audio and video streaming.

Two pieces of software were needed for the project. The first was a web server that would contain and serve the radio station web pages. The second, and more important, was the audio and video server. Progressive Networks makes media servers for UNIX, Windows NT, MAC and many other operating systems.

During July of 1997, a mock setup of the server was done to research the amount of bandwidth that would be needed. During this time, different configurations of web server and media server software were tested. At this point, it seemed that Progressive Networks Easy Start Server 4.0 would be the media server due to its widespread use and ability to run on a Microsoft NT Workstation. Progressive Networks also was offering the software for free which was attractive in trying to keep costs down.

This was an introductory server that could be downloaded free from the Progressive Networks web site. If the software is downloaded at no charge, no documentation or service is provided. The server will allow up to 60 people access to audio or video at one time providing the bandwidth is available. A second option is also available, which included software on CD-ROM, support for one year, documentation, and free upgrades of the server, including CD-ROM and documentation, for $995. The service and support package for the Easy Start Server 4.0 was proposed and eventually purchased. This option would turn out to be worth its weight in gold.

The media software does not have to run on the same machine as the web server, but for this application it would be best as, during games and live broadcasts, it is necessary to change links on the pages that run the audio. It also allows the radio station to have its own domain name. Netscape Enterprise Server had been tested and backed by Progressive Networks as compatible with their software. The Enterprise Server is also free to educational institutions.

During the testing, it became apparent that a powerful machine was needed to handle the task. Progressive Networks suggests that multiple machines be used for different parts of streaming media, but with a single powerful machine it is possible to eliminate the need for other machines. The proposals included an IBM-compatible Gateway machine. Due to the student's previous experience with Gateway, it seemed a reasonable choice, but any fast PC would be appropriate. Progressive Networks has minimum requirements for the machine, but flexibility and longevity were also important to sell the project to the administration.

Our requirements included a very fast processor, mass primary and secondary storage, as well as backup capabilities. The system would also need a high-quality sound card. As the proposal took shape, it was apparent this project would not be within the budget of the radio station or the Office of Technology. In order to receive funding, it needed to be proven to the administration that this would truly benefit the campus. A major justification for the project included the athletic department.

Support was not difficult to obtain from the coaches once they found out that their teams' games could be heard all over the world. The involvement of the athletic department and the fact that alumni would be able to tune in for sports broadcasts made the administration interested. This began to drive the project. Once the administration found out what this could mean to alumni, the possibilities of donations became apparent.

Three proposals were developed and presented to the administration for consideration. [Tab. 1] The first was a full package that included everything necessary for RealAudio and RealVideo. The second was a slightly scaled-down version that did not include items such as technical support for some of the packages. The third was a bare minimum that provided for audio, but no video.

The administration was very excited about the possibilities that the project could provide if it had the capability of both audio and video. The first proposal was accepted with a few modifications.

	Proposal 1	**Proposal 2**	**Proposal 3**	**Approved Package**
Processor:	Intel 300 MHz Pentium II Processor w/MMX	Intel 300 MHz Pentium II Processor w/MMX	Intel 300 MHz Pentium II Processor w/MMX	*Intel 300 MHz Pentium II Processor w/MMX*
Memory:	128 MB EDO DRAM	128 MB EDO DRAM	96 MB EDO DRAM	*96 MB EDO DRAM*
Cache:	Internal 512K L2 Secondary write-back cache	Internal 512K L2 Secondary write-back cache	Internal 512K L2 Secondary write-back cache	*Internal 512K L2 Secondary write-back cache*
Monitor:	Vivitron 17" color Monitor	EV 500 15" Color Monitor	EV 500 15" Color Monitor	*Use Sony Monitor already on campus*
Graphics Accelerator:	STB ViRGE[TM], 4MB, 3D 64-bit	STB ViRGE[TM], 4MB, 3D 64-bit	STB ViRGE[TM], 2MB, 3D 64-bit	*STB ViRGE[TM], 4MB, 3D 64-bit*
Hard Drive:	9GB SCSI Drive w/Adaptec 2940 PCI Controller	4GB SCSI Drive w/Adaptec 2940 PCI Controller	4GB SCSI Drive w/Adaptec 2940 PCI Controller	*9GB SCSI Drive w/Adaptec 2940 PCI Controller*
Floppy Drive:	3.5" 1.44 Drive	3.5" 1.44 Drive	3.5" 1.44 Drive	*3.5" 1.44 Drive*
CD-ROM:	12X Min./ 24X Max.	12X Min./ 24X Max.	12X Min./ 24X Max.	*12X Min./ 24X Max.*
Sound Card:	Ensoniq PCI wavetable sound card.	Ensoniq wavetable sound card.	Sound Blaster 16	*Ensoniq PCI wavetable sound card.*
Network Card:	IBM PCI card.	IBM PCI card.	IBM card.	*SN2000*
Operating System:	Windows NT Workstation	Windows NT Workstation	Windows NT Workstation	*Windows NT Workstation*
Service Program:	Gold Service and support 1year.	Gold Service and support 1year.	Gold Service and support 1year.	*Gold Service and support 1 year.*
Iomega Drive:	Internal Zip Drive.	Internal Zip Drive.	Internal Zip Drive.	*External Zip Drive*
Tape Backup:	TR4 SCSI TBU tape.	TR4 IDE TBU tape.	TR4 IDE TBU tape.	*TR4 SCSI TBU tape.*
Power Backup:	APC uninterrupted power supply 650	APC uninterrupted power supply 650	APC uninterrupted power supply 650	*APC uninterrupted power supply 650*
Total:	$5566	$4651	$4435	*$5145*

Table 1. Proposed Hardware and Approved Items

Several days after the proposals were presented to the administration, the project was approved. In September 1997, a combination of the proposal had been agreed upon, and funds were granted for the project. The hardware setup that was approved was at the high-end of what had been submitted and would be perfect for the job. [Tab. 1] The software package was approved and would include the $995 server and support pack for the Easy Start Server.

Installation

The computer to be used for the server was purchased from Gateway, which provides discounts to colleges. This helped cut costs once again. In early October, the Pentium II 300 MHz chip was still very new and in high demand. This was the only hold-up on the shipment of the computer. It took approximately two weeks to receive the machine.

It was determined that the best location for the server would be at the radio station. This way, the server would have direct access to an audio input source. The only requirement for input was that the audio be compressed before entering the sound card. With this in mind, three options were available for live broadcasting. First, it was possible to run the audio directly from the transmitter, but in our case that would mean drilling a hole through a thick concrete wall. The second option was to run audio off the studio distribution amplifier, through a compressor and then into the computer. The third was the easiest, but required the purchase of an audio tuner. By hooking the tuner up to the sound card, the signal coming out of the tuner and into the sound card was already compressed and would work well for broadcasting over the Internet. The only drawback to this is that it is not possible to do live broadcasting when the station is off air. The third option was chosen.

The original configuration of the computer was not a problem. From the manufacturer Windows NT Workstation 4.0 was already installed. Windows NT Workstation differs from Windows NT Server in many ways. First and most importantly, it is intended to have work done directly on the machine. Normally, with NT Server, the machine would be configured and then left alone, but with this application, that was not the case.

The installation of Netscape Enterprise Server was a simple process of downloading it from the Internet. The installation and use of the web server was one of the areas that had been tested during the summer of 1997. In the mock setup, it did not appear that the support package for the web server was needed.

The installation of the software from Progressive Networks was the most difficult. The software can be downloaded, or in this case, the CD-ROM was available. The installation took only about 30 minutes, but that did not include the configurations needed to make the software run. These configurations proved to be difficult, and created many situations in which trouble shooting was required.

Troubleshooting

The first item to be addressed was the basic setup of the computer. The computer had been purchased without a monitor and network card. Which were to be supplied by the Office of Technology. The original monitor intended for the server was a Compaq 15” color monitor. This did not work. For some reason the refresh rate of the monitor and the video card that came with the Gateway had slight problems. The screen would always appear to be wavy and after a few days, it was decided that the Compaq would be swapped with a Sony Trinitron. The monitors were changed and the problem cleared up.

In mid-October the project came to a halt. The network card that was used on the server, an SN 2000, was also used throughout campus. The SN2000 is a widely used and economically priced network card. In previous experiences this was a simple card to install, but once it was put into the machine a series of problems were encountered. The machine would no longer boot without producing many conflicting ports and IRQ address conflicts in the event log. It was apparent that some type of communication port conflict was taking place, but the specifics were still not clear.

The communication ports were reconfigured many times, but the problem still persisted. After five days of working on this problem, Gateway was contacted. Their service and support center was also confused. First, the network drivers were reloaded, but that was unsuccessful. Gateway also provided the latest upgrade to the driver, but it still was not configuring properly. The final analysis left the staff at Gateway to believe that it was a matter of an incorrect load of Windows NT Workstation. After reloading

Windows NT twice with no results, the hard drive was reformatted. Finally, the problem was solved and the network card was no longer causing conflicts. Gateway suggested that Windows NT had been incorrectly installed at the factory. By this time, it was nearing the end of October and the latter part of the football season.

As mentioned earlier, one of the big selling points for this project was the fact that alumni would be able to listen to athletic events, including football. The bad (and good) news was that the football team was undefeated, and alumni were wondering when they would be able to tune in. The project completion was set for the Homecoming game on November 1.

Once access to the network was achieved, the installation of the web server was the next task. The Netscape Enterprise server was downloaded from their site. Installing the web server was very easy, as Enterprise Server needs only a few bits of information about specifics. Once it knew the IP number of the machine, and it was verified that it was able to listen to port 80, all that had to be done was load the radio station homepage and rename it to be the index page. The Office of Technology had already registered an IP address for the station under “webcast.doane.edu.”

Once the web server was up and running, it seemed as though it would not be long before audio and video was streaming across the network. However, installation of the Progressive Networks Easy Start server had some complications. The first of these was the configuration of the IP binding that was needed to run both the web server and the media server on the same machine.

In most of the information that is provided by Progressive Networks, it was suggested that the web server and media server run on separate machines. The reasons for the two machines is that a much slower and less powerful machine can handle each individual task, but a Pentium 300 could handle the multiple tasks. How the software puts together streaming media is an important consideration.

The software is made up of four parts. Progressive Networks suggests that each of these four parts reside on a separate machine. First is the media server. The media server listens to the assigned IP address and then feeds out the streams that are requested. The second is a web server, which provides the location for web pages that contain links that users follow to choose what type of media they want. Examples of this might be an archived file of a Christmas concert, sports event, or the live broadcasts. The third component is the encoder. The encoder takes either files or live input and connects with the media server so that it can stream the media out to users that have requested it from web pages. The fourth element is the system monitor. The system monitor does not have to run, but it helps track who is connected to the media server, how many people are connected, and what kind of bandwidth is being used on the network. The system monitor is very taxing to the machine on which it is running, and should not be on at all times, or should be set only to update at limited intervals and not continuously. [Fig. 1]

Progressive Networks has a recommended setup, but it is not the only possibility. [Fig. 1]

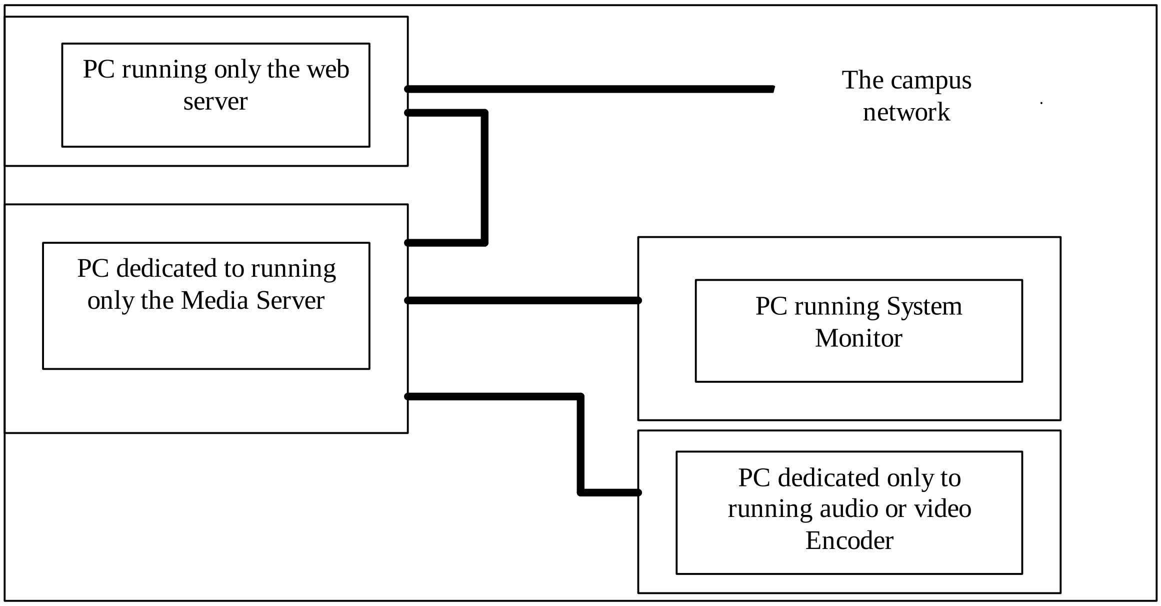

PC running only the web server
The campus network
PC dedicated to running only the Media Server
PC running System Monitor
PC dedicated only to running audio or video Encoder

Figure 1. The suggested setup from Progressive Networks using multiple PCs

It was decided that only one machine would be used for the project so that everything could be done at one PC in one location. [Fig. 2]

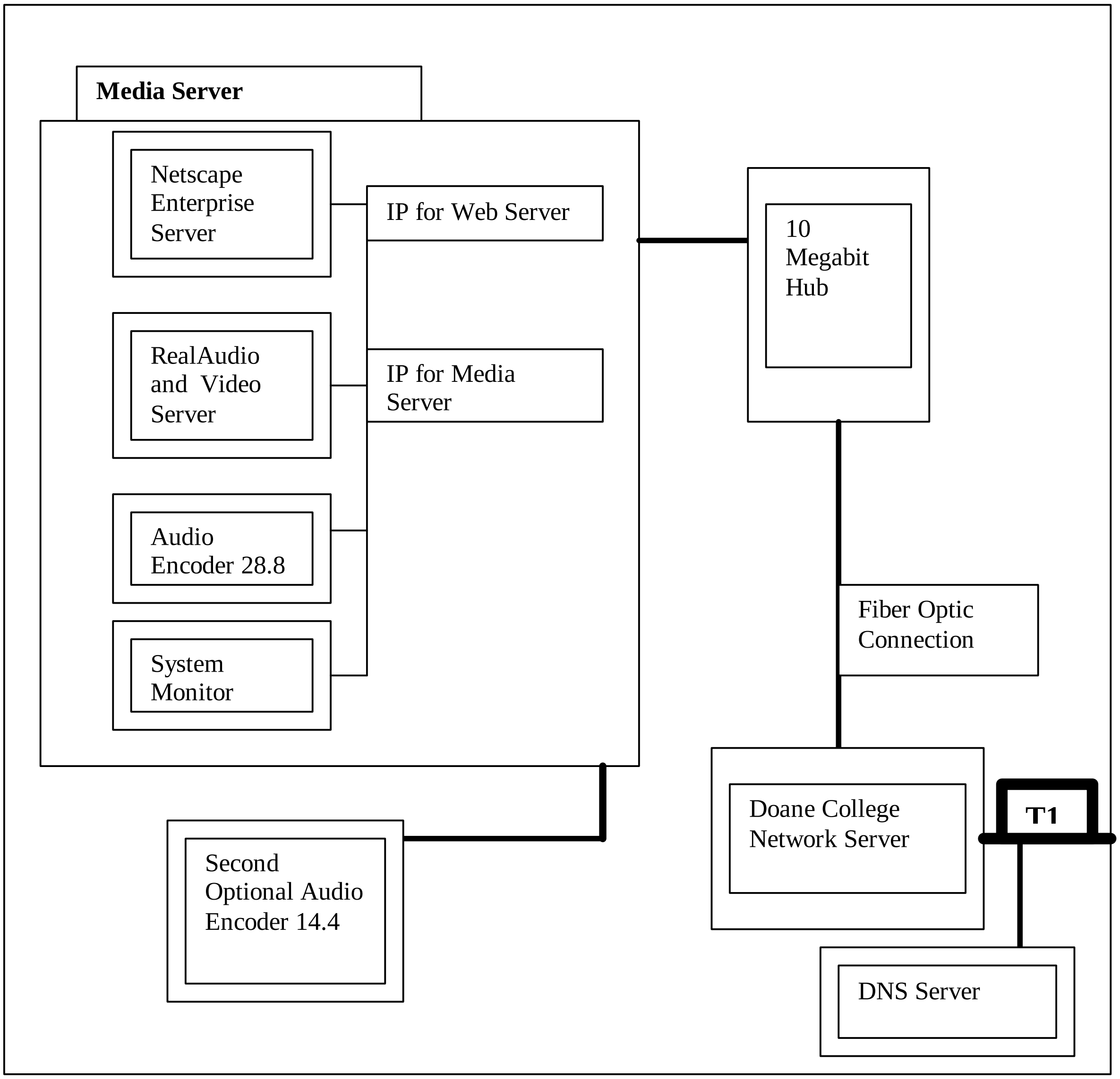


Figure 2. Doane College Media Server Setup

It is important to notice there is one physical connection to the hub from the machine, yet two IP addresses are needed: one for the web server and one for the media server. [Fig. 2] IP binding, an element that is provided by Windows NT, allows for this.

The domain name that the web server uses is registered as “webcast.doane.edu.” The server listens to port 80 of the IP address that it was given. The one problem with this is that Progressive Networks' media server also must listen to port 80. Within the Windows NT network configuration, it is possible to set up two IP addresses for one machine. Once the machine has two IP addresses, it can be assigned to listen to port 80 of the second IP address, thus allowing the two servers to run on one machine. The only complication of this process was in the configuration files that the media server uses. A file called the config.sys contains almost all of the important configurations that the server needs. Unfortunately, documentation on how to setup the configurations was not accurate. After trying many different settings for this file, Progressive Networks was contacted for assistance. After talking with the technical support people at Progressive Networks, the problem was resolved, and the media and web servers were no longer conflicting with the same IP address. The problem was that a set of brackets had been omitted from the documentation.

Once both the web server and media server were running, it was a matter of testing and modifying as necessary to make sure that everything worked. Progressive Networks had recently added a new file extension to the audio files, which meant setting up a new MIME type on the web server. The system manager that Netscape has built into the server makes it easy to add MIME types.

Everything was now working and ready to go live. It was Thursday and the first official football game was to go over the Internet on Saturday. With everything in place, testing of audio files began. Progressive Networks sends test files for the audio and video on the CD-ROM that comes with the software. Audio and video can be encoded in what are called codecs. These are different compression rates that allow for the audio or video to be viewed at a better quality depending on what type of connection the user has. This can range from poor quality over a 14.4 KBPS to almost CD Stereo quality over ISDN or LAN-type connections. For our purposes we only encode for the basic 28.8 KBPS modem users. On Friday of that week, the first live test files were working properly, and the radio station had been receiving calls from around the country for individuals interested in tuning in.

On Saturday, November 1, 1997 the broadcast of the Homecoming game went off without a hitch. During the game, the server saw as many as 150 people log on. The maximum number of users at any given time was 25. After the game positive, feedback came from alumni and parents. This feedback was passed on to the administration and athletic department. The football team continued to play out the rest of the season, creating a high demand for RealAudio during the playoff games.

During the final two playoff games, the server did not hit the maximum of 60 users at one time, but came close with 53. The quarterfinal playoff game brought almost 450 listeners to the site. It became a contest at the radio station to see from where the furthest user tuned in. Although people tuned in from all over the country, the winner of the contest was a friend of the football team in New Zealand. He had listened to the final game live from the Doane College web site a half a world away. This was just one of the many exciting stories that came with the media server.

One problem that arose was that everytime a live broadcast took place, someone had to be at the server to start and stop the programming. The solution came with the availability of what is called batch processing. Batch processing allows the encoders to be started and stopped by batch files. The creation of the batch files is not easy, as the documentation is very vague. Once in place, however, it is a simple matter of scheduling the batch files to run from within Windows NT.

Not long after everything seemed to be running beautifully, the inevitable happened. Progressive Networks made an update to the server software that fixed many of the previous bugs. The new server was called the Basic Server 5.0 allowing for better quality on slower connections. An upgrade was planned to take place during the Christmas break in late December. During the break, the radio station would be off the air and hopefully the upgrade would go smoothly.

As should have been expected, the upgrade was anything but smooth. Progressive Networks suggested that the current server not be uninstalled until the new server is running. However, this may not be the best approach. Parts of the existing 4.0 server conflicted with the 5.0 server and would not allow it to run. The reinstall of the 5.0 was attempted three times before it was decided to uninstall 4.0 completely and start all over. Progressive Networks does not suggest this, but it proved to be the easiest way.

The new server would no longer be able to use the batch files that had been used for encoding on 4.0, so new files were written. Once again, the files had to be tested and modified several times before they would work. Running the encoder off of batch files is tricky, but in the long run it is well worth the effort.

In mid-January, the project was back on course. A week after this the station was contacted by Progressive Networks via e-mail and notified that a bug had been found in the 5.0 software. The bug would allow a hacker to get into the server and shut it down from a remote location over the Internet. The solution for this latest problem was Basic Server 5.01. This upgrade went much more smoothly than the change from 4.0 to 5.0.

Summary

The project has been a big success on campus. One of the reasons for this was the fact that it brought many departments together to do something not possible by just one. The main focus of the project at Doane College has been audio, but in the future the college will be using RealVideo to stream the college's promotional tape over the Internet. The project has been used to archive speeches that have taken place on campus as well as concerts from the music department. Most colleges offer static information via a web site. However, offering parents, alumni and friends of the college the opportunity to listen and see concerts, contests, speakers, as well as other events adds a new dimension to the web site that benefits not only those tuning in, but the college itself.

Acknowledgments

Special thanks to all the people who made this project possible. Without the help my family, Pam Kunzman, Dan Schneider, Dr. Alec Engebretson, J.S. Engebretson, Dr. Dan Kaiser and Dr. Lee Thomas, this would not have been possible.

Design Patterns In Java: *Sutherland*, A Software Framework for Evolutionary Computation

Nicholas Freitag McPhee Nicholas J. Hopper
Mitchell L. Reierson
Division of Science and Mathematics
University of Minnesota, Morris
Morris, MN USA-56267
(320) 589-6300
{mcphee, hoppernj, reiersml}@mrs.umn.edu
http://www.mrs.umn.edu/{~mcphee, ~hoppernj}

Abstract

In this paper we describe an object-oriented framework for Evolutionary Computation. We first describe the machine-learning paradigm known as Evolutionary Computation (EC) by way of two examples; A similar primer on the concepts of design patterns and frameworks follows. The main components of our framework are then laid out, and specific components for genetic programming are described. We conclude with a summary of the benefits of working in Java and indicate future work to be done toward the ultimate goal of creating a flexible and extensible library of “off-the-shelf” components which can be combined to perform nearly any EC task.

Design Patterns In Java: *Sutherland*, A Software Framework for Evolutionary Computation

Nicholas Freitag McPhee Nicholas J. Hopper
Mitchell L. Reierson
Division of Science and Mathematics
University of Minnesota, Morris
Morris, MN USA-56267
(320) 589-6300
{mcphee, hoppernj, reiersml}@mrs.umn.edu
http://www.mrs.umn.edu/{˜mcphee, ˜hoppernj}

Abstract

In this paper we describe an object-oriented framework for Evolutionary Computation. We first describe the machine-learning paradigm known as Evolutionary Computation (EC) by way of two examples; A similar primer on the concepts of design patterns and frameworks follows. The main components of our framework are then laid out, and specific components for genetic programming are described. We conclude with a summary of the benefits of working in Java and indicate future work to be done toward the ultimate goal of creating a flexible and extensible library of "off-the-shelf" components which can be combined to perform nearly any EC task.

1 Introduction

Evolutionary Computation (EC) is a machine-learning paradigm which performs a near-optimal search of the space of solutions to a given problem by applying the Darwinian principles of fitness-proportionate reproduction and genetic crossover to a population of solution candidates. There are currently several EC toolkits available which use traditional methods of software construction; most of them are too specific to easily allow for reconfiguration of the system for different problems or problem types.

In this paper we describe an alternative EC system that utilizes a variety of object-oriented programming techniques to provide a more flexible set of tools. Particularly important are two key ideas from recent research in software design: Frameworks and Design Patterns. A framework is a set of cooperating classes that make up a reusable design for a specific class of software, while a design pattern systematically names, explains, and evaluates an important and recurring design in object-oriented systems.

Sutherland applies the principles of Design Patterns and object-orientation to construct a Framework of extensible and reusable components. Our goal is that these components can be used like ”off-the-shelf” parts that can be combined to perform nearly any EC task. The design also allows other researchers to extend the framework by adding new components or expanding the functionality of existing components in a consistent manner.

After initially working in C++, we chose to implement *Sutherland* in Java which provides several advantages. The Java Virtual Machine provides platform independence and automatic memory management. Java's standard libraries also make it easy to implement reliable distributed computation and remote configuration and observation of experiments, as well as providing concrete implementations of several design patterns.

2 Evolutionary Computation

Evolutionary Computation (EC) is an approach to problem solving based on concepts borrowed (often loosely) from biological evolution. There are many varieties of EC, but they all share a common structure. In each case one develops some means of representing potential solutions, and then casts the problem as one of developing or searching for the desired solution in the space of possibilities defined by the representation. Instead of using standard Artificial Intelligence search techniques, however, EC uses simulated evolution to perform the search. An initial population of randomly generated individuals is built, and each individual is assigned a fitness that is a measure of how well they solve the problem at hand. The best individuals are then mutated or combined to generate new individuals, some of which are (hopefully) better than their "parents". This process continues until an acceptable solution is found, until a predetermined maximum number of generations have passed, or until some other termination criteria is met.

To make this more concrete we'll present two simple examples (function optimization and symbolic regression) in some detail. We'll then review the common components of most EC algorithms with an eye to identifying components and patterns that will be important in the design of *Sutherland*.

2.1 A GA example: Function optimization

Our first example is that of function optimization. Imagine, for example, that we have a function f of several arguments, and we want to find values of the arguments that maximize (or minimize) $f(a, b, c, \ldots)$. While f might be defined explicitly, it's common for f to be defined implicitly. The parameters $(a, b, c, \ldots)$ could, for example, define a strategy in a game playing environment, and $f(a, b, c, \ldots)$ could be implicitly defined as the score the strategy defined by $(a, b, c, \ldots)$ receives when matched with a set of other strategies. As another example, the parameters could define key aspects of an airplane wing design, and the function f might be implicitly defined as the performance of the design described by the parameters in a simulated wind tunnel.

In the remainder of this section we'll use the following simple example: Given a 1024×1024 lattice, find a set of three lattice points that maximizes the sum of the (Euclidean) distance between the three pairs of points. (This is a fairly simple problem to solve analytically, but there are quite difficult variants of it, e.g., the problem of choosing five points on the surface of the unit sphere that maximizes the sum of the 10 distances they define.) Here we can define our parameters to be the (x, y) lattice coordinates of each of the three points. Thus we can take f to be a function of six coordinates a_0, a_1, …, a_5, and the problem becomes one of maximizing $f(a_0, \ldots, a_5)$ for a_i in the range $0 \leq a_i < 1024$.

The first task in most EC applications is to determine an appropriate representation for candidate solutions. In our optimization problem the parameters are numeric, and a simple representation for a candidate solution is a binary string, where different substrings are the binary representation of the values of different parameters. Since our six parameters are limited to the range $0 \leq a_i < 1024$, 10 bits is sufficient to describe all the legal values of a single parameter. Thus 60 bits suffice to describe all the legal combinations of values for our six parameters. We can, therefore, take a bitstring of length 60 as a representation of a candidate solution, with the first ten bits representing the x coordinate of the first point, the next ten representing the y coordinate of the first point, etc.

This fixed length bitstring representation is the standard representation in Genetic Algorithms (or GAs), and dates back to John Holland's initial work on GA's in the early 1960's [Holland 1962]. GAs have been studied quite extensively and while the representation does have a number of limitations, they also have a variety of nice properties; see any of [Holland 1992, Goldberg 1989, Mitchell 1996] for more information.

Given this GA representation we can easily generate an initial population of randomly generated individuals for our example problem by generating random bitstrings of length 60. Each of these randomly generated bitstrings has a fitness which is defined to be the sum of the three distances determined by that bitstring. We would expect some of these randomly generated strings to be more fit than others, i.e., to have higher total distances. We then use some selection mechanism to select these more fit individuals, and mutate or recombine them in some way to generate new individuals.

The two main recombination operators in GAs are a point mutation operator and a crossover operator. Both of these have a number of variants, but to keep things simple we'll only present one variant of each here. The

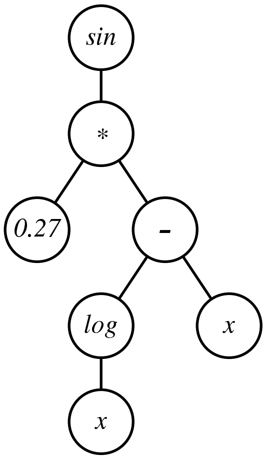


Figure 1: The parse tree representation of $f(x) = \sin(0.27 * (\log(x) - x))$.

point mutation operator takes a bitstring and randomly flips a bit to get a new bitstring. The crossover operator takes two bitstrings $[w_0, w_1, \ldots, w_{n-1}]$ and $[z_0, z_1, \ldots, z_{n-1}]$, and generates a new bitstring by replacing a section of the first with the corresponding section from the second. In two point crossover, for example, we randomly choose two indices i and j in the interval $[0, n)$ and we take i and j to define the interval in u to replace with the corresponding interval in v. If $i \leq j$ the resulting bitstring is

$$w_0, w_1, \ldots, w_{i-1}, z_i, z_{i+1}, \ldots, z_{j-1}, w_j, w_{j+1}, \ldots, w_{n-1}$$

and if $i \geq j$ the resulting bitstring is

$$z_0, z_1, \ldots, z_{i-1}, w_i, w_{i+1}, \ldots, w_{j-1}, z_j, z_{j+1}, \ldots, z_{n-1}.$$

We then update the population by inserting these new individuals (typically in place of some or all of the "older" individuals). This process is repeated until we reach some pre-determined goal, we reach a pre-determined generation limit, or until the population converges to multiple copies of a single individual.

2.2 A GP example: Symbolic regression

As our second example, we'll consider the problem of finding a function that describes a given set of data. Imagine we have a set of data points $\{(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})\}$, and we wish to find some function f that (nearly) passes through those points, i.e., $f(x_i)$ is (nearly) equal to y_i for $0 \leq i < n$. The points might be data generated in a lab experiment, and our goal might be to find some function that describes that data, either in an effort to better understand the data, or in an effort to predict future values in the series.

Here it seems unlikely that we can (easily) represent candidate functions with GA bitstrings as in the previous example. A more natural representation might be parse trees. We could define a set of functions, constants, and variables to use as building blocks, and define a candidate function to be a (syntactically and semantically legal) parse tree built from those components. We might, for example, define our set of functions to be $\sin, \cos, \log, \exp, +, -, *, /$, our set of constants to be a collection of randomly generated real numbers in the range $[0, 1)$, and our set of variables to be the single argument x. Then parse trees built out of these components could represent candidates for a function matching a collection of data points as described above. The tree in Figure 1, for example, represents the function

$$f(x) = \sin(0.27 * (\log(x) - x)).$$

This parse tree representation is the standard representation in Genetic Programming (or GP), largely due to John Koza [Koza 1992]. While much newer than most other EC variants (such as GAs), GP has proved very successful in a variety of application areas. GP is typically more computationally intensive than GAs, but the flexibility of the parse tree representation makes it much easier to develop “natural” representations of potential solutions for complex problems.

Given this representation we can generate an initial population of random parse trees, and assign a fitness to each of these by, for example, computing the sum of the squared error of the candidate function on each of the test points.

Once we’ve assigned fitnesses we can select the more fit individuals and and mutate and recombine them to generate new (hopefully more fit) individuals. There are a variety of potential mutation operators, including the simple approach of replacing some randomly chosen subtree with a new randomly generated subtree. (For a discussion of a very interesting collection of GP mutation operators based on the notion of minimum edit distance see [O’Reilly and Oppacher 1994].) The standard GP recombination operator is a crossover operator which takes two “parent” trees and generates a “child” tree by choosing a subtree at random from the first parent, and replacing it with a subtree chosen at random from the second parent.

2.3 Key EC components

This brief review of two different approaches suggests the basic EC algorithm, and indicates several major components (both algorithms and data structures) that one would want to be reflected fairly directly in a framework such as *Sutherland*. Among these are

Representation A data structure used to store and manipulate the representation of a potential solution.

Fitness A data structure which represents the fitness of an individual. Fitnesses will need to support basic comparison operations, as well as statistical operations such as averaging.

Individual A data structure which associates a Representation with other important “bookkeeping” information such as that individual’s Fitness.

Evaluation function An algorithm that maps Representations into Fitnesses.

Population A data structure that is essentially just a collection of Individuals.

Selection operator An algorithm that chooses one or more Individuals from a Population based on some criteria.

Recombination operator An algorithm that takes one or more Representations and generates a new Representation.

Top-level control loop An algorithm that encapsulates the key commonalities of most EC algorithms, e.g., building an initial population, selecting individuals for recombination, etc.

One of the key goals in designing *Sutherland* was to separate representation independent concepts from those that depend on a particular representation. As an example, we saw that both GAs and GP have notions of mutation and crossover, or, more generally, recombination operators, that are independent of the specifics of bitstrings or parse trees. The *particular* operators (two point crossover on bitstrings vs. parse tree crossover) are clearly dependent on the representation, but the *concept* of a crossover operator is not. In our framework we reflect this by having a representation independent notion of crossover with particular instances that are (by necessity) specific to particular representations.

As we shall see in Section 4, many of these problems are easily dealt with using the inheritance mechanisms provided by most object-oriented programming languages. Some, however, are more subtle, and require the use of some of the more advanced patterns discussed in Section 3.

3 Frameworks and Patterns

In order to accomplish our goals of extensibility and reuse, *Sutherland* relies on several recent ideas from object-oriented software design. The two most important of these ideas are Frameworks and Patterns. A design pattern is an abstract name and description for a common, recurring structure in an object-oriented design. A framework is a collection of cooperating classes which can be assembled to form an application for a specific problem domain; through interface and implementation inheritance a framework can be used to produce a new application from the same domain with little new design or implementation work.

3.1 Design Patterns

A design pattern is a high-level description of a class or interaction between classes in an object-oriented design. Each design pattern names, abstracts and identifies the key aspects of a common design structure that make it useful for creating a reusable object-oriented design [Gamma *et al.* 1995]. By using patterns, we hope to make it easier for others to understand, extend and reuse *Sutherland*.

Design Patterns generally fall into one of three categories. Creational patterns specify methods of creating new objects without coupling code to a specific class. Structural patterns specify how objects and classes can be composed. Behavioral patterns specify the interactions and responsibility distributions between objects and classes. *Sutherland* includes examples of all three types of patterns; Here we will provide an example of each category.

3.1.1 Behavioral Patterns: The Iterator

Consider the notion of a container object. There are many different algorithms and data structures for implementing an object which itself contains a collection of objects. Examples include singly- and doubly- linked lists, arrays, deques, queues, stacks, sets, trees, and hash tables. Although the access methods for each of these types of containers are different - that is, we access elements in an array with an index, in a hash table by a key, and in a stack by position - there are many algorithms which can be made to use any container by accessing its elements one at a time. An object which provides serial access to the elements of a container is an example of the iterator pattern. The essential elements of this pattern are simple: the iterator is created by the container object upon request; provides a way to access the current element, move to the next element, and determine if the last element in the collection has been seen; and the iterator relieves the container object of responsibility for traversal. The Standard Template Library of C++ provides one example of the notion of an iterator; the Enumeration interface from Java's standard library provides another. In both cases, the interface of the iterator reveals nothing about the structure of the underlying container, and some container objects might provide several different iterators for different types of access - for example, moving forward or backward through the elements of a list. In all cases, making iterators responsible for serial access instead of directly accessing the underlying containers allows one to change between different container types or implementations without modifying the code that accesses them.

In *Sutherland*, it is necessary in several places to manipulate parse trees for the computer programs being evolved. Because several implementations of parse trees could be used, objects which need to access all of the elements of a parse tree were designed to use a parse tree iterator object instead of directly accessing the tree itself. *Sutherland* specifies two kinds of parse tree iterators corresponding to two common kinds of tree traversals applicable to k-ary trees, preorder and postorder. By abstracting the process of tree traversal, we are able to implement many components without tying them to a particular tree representation, making the design more modular and greatly increasing the opportunities for code re-use.

3.1.2 Structural Patterns: The Bridge

The Bridge pattern decouples an abstraction from its representation, allowing the two to vary independently. In this fashion, code that is concerned only with the abstraction is not tied to the implementation, and vice versa. An example of a Bridge is the Chromosome class described in Section 4.

3.1.3 Creational Patterns: Abstract Factory

The Abstract Factory pattern provides an interface for instantiating families of related objects without coupling code to their specific classes. For example, in the code in Figure 2, the first version of the ShapeCollection class is tied to the specific shape classes known of at the time of its creation; the second version allows

```
Class ShapeCollection {             | Class ShapeCollection {
   ...                              |  ...
   void AddNewRectangle(Color c) {  |  void AddNewShape(ShapeMaker sm) {
      Shape s = new Rectangle(c);   |     Shape s = sm.NewShape();
      ...                           |     ...
   }                                |  }
   void AddNewCircle(Color c) {     |  ...
      Shape s = new Circle(c);      | }
      ...
   }
   ...
}
```

Figure 2: Application of the Abstract Factory pattern

the class to be more flexible in adapting to future shape classes.

In *Sutherland*, a general population class needs a way to create new individuals without being coupled to a specific type of individual. The Rep_factory interface discussed in Section 4 provides this mechanism. This decouples our code from one specific representation of an individual; in fact we have performed experiments using two different representations with *Sutherland* - see [McPhee *et al.* n.d.*b*].

3.1.4 Other Patterns used in Sutherland

In addition to iterators, bridges, and abstract factories, *Sutherland* incorporates a variety of other important patterns. Examples include:

- Observer represents a relationship which allows an object to notify all of its dependent objects when its state changes
- The Visitor pattern allows the performance of various operations on the elements of an object structure without changing the classes of the elements on which it operates.
- The Strategy and Template Method patterns allow the modification of algorithms independently of the clients using them.
- The Singleton pattern ensures a class has only one instance.

Each of these patterns contributes in a significant way to the extensibility and modularity of *Sutherland*; several are discussed further in Section 4.

3.2 Frameworks

While design patterns describe the interaction between small groups of classes or objects, frameworks describe the overall architecture of an application. A Framework is a set of cooperating classes that make up a reusable design for a specific class of software [Gamma *et al.* 1995]. For example, the Microsoft Foundation Classes (MFC) framework provides a set of classes for designing a Microsoft Windows GUI application; another framework might provide a set of classes to create compilers for different programming languages and target architectures. An application programmer customizes a framework for his or her application by creating application-specific subclasses of the classes and interfaces defined in the framework; a good framework design should minimize the amount of subclassing necessary while remaining flexible enough for a wide variety of applications.

By specifying the key classes, their interaction and thread of control, a framework capitalizes on the similarity in design problems for a given application domain, and thus minimizes the amount of (re)design that goes

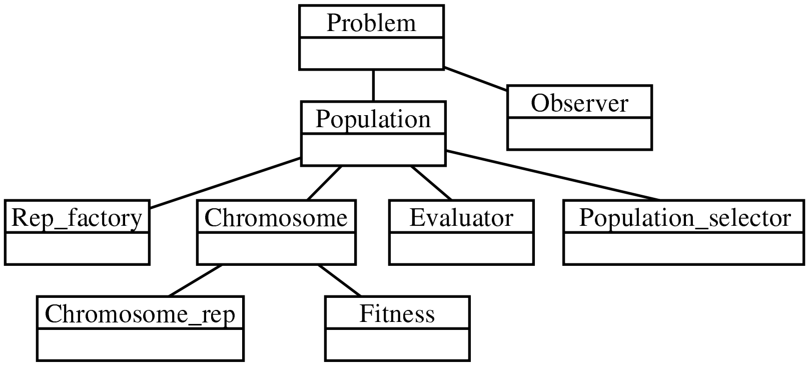


Figure 3: The primary use relationships between the key components of *Sutherland*.

into application development. The major difference between, for example, a framework and a toolkit or library is that while using a library entails writing the main body of the application and calling the code you wish to reuse, using a framework entails reusing the body of the application and (re)writing the code that it calls [Gamma *et al.* 1995]. Coupled with working implementations of several of the interfaces it specifies, a framework can considerably speed application development while at the same time making the software more flexible and reusable; the major disadvantage of using a framework is the loss of design control, since many of the decisions involved with the design have been made already.

Because a framework must address common design issues, and because documentation is critical, frameworks are most effective when paired with patterns. The use of patterns and the documentation of their use will help others familiar with patterns understand the interactions more quickly, and the uniform structure imposed by design patterns makes the architecture easier to understand even for those with no prior exposure to patterns. Such a combination helps to make the framework not only more extensible, but easier to understand, which in turns makes development with the framework that much easier.

4 General Sutherland components

In this section we will discuss some of the major components of *Sutherland* that are independent of any particular EC domain or problem. In the following section we'll discuss how we implement Genetic Programming by providing particular implementations (via subclassing) of several of these components.

The major components and their "use" relationships are illustrated in Figure 3, and will be discussed below.

4.1 The `Problem` class

The `Problem` class encapsulates the problem-specific components of an EC run. These include

- The choice of chromosome representation factory. This encapsulates, for example, the choice of representation and the associated recombination operators.
- The evaluator, which will map chromosome representations to Fitness values.
- The selection mechanism, used to choose individuals to use as parents and to choose individuals to replace.
- The population size and several other problem specific parameters.

The other important function of the `Problem` class is that it is *observable*, i.e., inherits from Java's standard library `Observable` class. This allows one to attach *observers* (classes inheriting from Java's `Observer` class) to a Problem; the role and importance of observers is discussed in Section 4.2.

4.2 The Observer/Observable pattern

As discussed above, the Problem class inherits from The Observable class in Java's standard library [Gosling *et al.* 1996]. This allows one to attach observers (clases inheriting from Java's Observer class to a Problem. The observer/observable pattern [Gamma *et al.* 1995] is crucial in separating the execution of an EC run and its output. This allows one to change the kinds of output that is generated without modifying any framework classes.

The basic mechanism of the pattern is that the observed class (Problem) notifies whatever observers have been attached at regular intervals (e.g., every generation). When notified, an observer can then query the Problem class and generate whatever sort of output is defined for that observer.

Sutherland currently supports several different observers, each of which serves a different purpose:

- Generating summary textual output which is then sent to the standard output so users can track the general progress of the run.
- Generating summary textual output which is then sent out through a telnet port so users can remotely track the general progress of the run from anywhere on the network. (Several components in Java's standard libraries make implementation of such an observer extremely easy.)
- Generating detailed textual output which is stored in one or more files, providing a detailed trace of the run and its results.

In the future we plan to extend this to include several new observers, with special emphasis on network observation. Java's network and web tools, for example, make it quite feasible to implement observers that would allow one to interactively observe a remote run via a web page, changing on the fly the type and quantity of data presented.

4.3 The Population class

This class is essentially a wrapper around a container class used to hold the individuals in a population. It also encapsulates the loop that selects the best individuals, recombines them, and adds the results back into the population.

4.4 The Chromosome/Chromosome_rep pattern

The Chromosome class collects together a an individual's representation along with other information such as fitness. There are two major ways one could handle the need to have different representations (such as bitstrings for GAs and parse trees for GP):

1. Define Chromosome as an abstract base class, and then subclass it for each different representation. Each subclass would then define a particular representation, such as bitstrings or parse strings.
2. Make a separate Chromosme_rep class, and define Chromosome to hold an instance of Chromosome_rep.

We took the second approach because it better separates two logically independent issues: The chromosome representation (e.g., the bitstring or the parse tree) and the book keeping information in the chromosome. Probably the most important advantage this provides is the ability to create a variety of chromosomes (e.g., ones that keep track of who their parents are, and ones that don't) and a variety of representations (e.g., bitstrings or parse trees) without getting an explosion of classes. In the first approach every combination of a variety of chromosome and a variety of representation would require a new class, generating $m * n$ classes for m chromosome varieties and n representations. Taking the second approach prevents this explosion by allowing one to freely combine chromosomes and representations, requiring only $m + n$ classes.

This is an example of a *Bridge* pattern [Gamma *et al.* 1995].

4.5 The `Rep_factory` pattern

The standard mechanism for creating instances of a class is through constructor methods in a class. This, however, requires the code that creates instances commit to a particular (sub)class and constructor. This, much like in the previous example, tends to cause a rapid multiplication of classes.

An alternative which we use in *Sutherland* is to have a separate *factory* class whose sole job is to build instances of the desired class. In *Sutherland*, for example, we had a `Rep_factory` class which builds instances of `Chromosome_rep`. This approach allows classes like `Population` to refer only to the `Rep_factory` base class. Then instead of subclassing `Population` to modify which representation is used, we merely have to pass in a different `Rep_factory`.

This is an example of the *Abstract Factory* patttern [Gamma *et al.* 1995].

4.6 The `Evaluator` interface

`Evaluator` provides an abstract interface for evaluators which map `Chromosome`s to `Fitness`es. All the major components (such as `Population`) refer to this abstract interface. This allows us to implement and use problem specific evaluation functions without affecting any of the code except for the problem specific instance of the `Problem` class.

4.7 The `Population_selector` class

The abstract `Population_selector` class defines an interface for tools that select individuals in a population, either to use as parents, or to replace. There are a variety of selection mechanisms used in EC, including tournament selection and fitness proportionate selection (see, e.g., [Mitchell 1996]). Having an abstract `Population_selector` class allows us to mix and match selection mechanisms simply by passing in different selection mechanisms as arguments to the `Population` class.

4.8 The `Fitness` class

The abstract `Fitness` class defines an interface for individuals' fitnesses. Fitnesses are required to support both a notion of $<$ and $=$; standard specific instances of fitnesses include `Int_fitness`, used to hold integer valued fitnesses, and `Real_fitness`, used to hold real valued fitnesses.

`Fitness` then has an abstract subclass `Additive_fitness` which defines an interface that supports operations necessary to compute averages and standard deviations.

One particularly valuable subclass of `Additive_fitness` is `Pair_fitness`, which allows users to build up structured fitnesses in a uniform way. A `Pair_fitness` collects two other `Additive_fitness`es together into a single fitness. The `Pair_fitness` then uses standard notions of pair ordering and pair equality, and uses the underlying notions of ordering and equality on the components. This allows the the user to build up arbitrarily complex structured fitnesses in the `Evaluator` class without having to define any new classes or alter any of the code that might use that Fitness (e.g., `Population_selector`).

5 GP Components

The Genetic Programming subclasses of the `Chromosome_rep` and `Rep_factory` produce some opportunities for application of additional patterns and reuse. The two relevent classes are `GP_tree_factory` and `GP_tree`, which are classes implementing `Rep_factory` and `Chromosome_rep`, respectively.

5.1 GP_Tree: Chromosome_Rep

The GP components of *Sutherland* subclass the `Chromosome_Rep` class with a parse tree representation. The parse tree class defines a basic interface for a node; specific types of parse tree nodes are in turn created by subclassing this generic node class. Its basic functions are:

- creating iterators, as discussed in section 3

- evaluating a tree; a virtual method is included in the base class allowing function nodes to evaluate their subtrees and apply a function to the values returned
- writing the tree to an output stream; here again subtrees can be printed out from function nodes by calling the virtual function defined in the base class
- obtaining the n^{th} specific subtree
- replacing the n^{th} specific subtree

Implementing these as `Chromosome_rep`s allows their implementation to vary independently of the Chromosome class; Multiple types of Chromosomes might wish to use `GP_tree`s as their `Chromosome_rep`. This is one area of *Sutherland* we wish to slightly change; as it stands our parse tree class is tied to one specific implementation of a parse tree and we feel that the framework could benefit from a more general parse tree class (which would not necessarily implement the `Chromosome_rep` interface).

5.2 Factories

There are several well-known algorithms for generating parse trees for GP; each is provided as a class with *Sutherland*. This allows developers to change the method of tree generation by simply changing which factory is passed as an argument to the Population in the Problem construction. The three algorithms are *grow*, which randomly "grows" subtrees up to a maximum depth; *full*, which makes full trees of a specified depth; and *ramped-half-and-half*, which uses *grow* on half of the trees and *full* on half. By composing a grow and a full factory, a ramped-half-and-half factory minimizes the amount of new code generated to create trees.

6 Benefits of Java

Initially we began development of *Sutherland* in C++, but current work on the framework is in Java, which provides many advantages over C++. Among these advantages are automatic memory management (also referred to as garbage collection), platform independence, cleaner multiple inheritance semantics, convenient creation of modules, and convenient standard libraries (especially for network components).

Java's garbage collection facilities make it easy to implement the GP technique known as tree sharing. The parse trees used to represent individuals in GP can share common subtrees, in effect implementing the entire population as one directed acyclic graph (DAG). This technique drastically reduces the memory footprint of a running GP problem; in our experience the memory footprint of a run which uses tree sharing stays roughly constant over the life of the run, while pressures such as reproduction accuracy cause the memory footprint to grow dramatically without such sharing [McPhee and Miller 1997].

Since Java compiles to platform-independent bytecodes, compiled Java can be run on a wide variety of operating systems and architectures. This along with the small memory footprint afforded by tree sharing makes it easy and practical for *Sutherland* to be used on a very wide variety of machines. Compared with other systems such as C++ which requires system-dependent makefiles and possibly system-dependent libraries, this makes *Sutherland* more flexible than most GP systems.

Java's "interface" mechanism provides a conceptually cleaner model of multiple inheritance than most Object-Oriented Programming Languages (OOPLs). In Java, a virtual base class with no instance variables and no instance methods can be declared as an 'interface' instead of an abstract class. Inheritance is divided into two types: the `extends` indicates implementation inheritance, as in C++ or Smalltalk, while the `implements` keyword indicates interface inheritance, where a class is guaranteed to implement all of the methods in the interface it inherits from. A class may `extend` only one other class, but may `implement` multiple interfaces. This removes most of the semantic complications introduced by general multiple inheritance, while still allowing a richer type system than single inheritance. The use of interfaces rather than abstract classes in several components of *Sutherland* allows implementation inheritance to be used in extending the functionality of non-*Sutherland*-specific classes in certain problems and representations (such as a tree or selector class) while keeping the type system of various other components more general.

The "package" mechanism built in to Java provides easy creation of module-like entities. By storing their source in a separate directory and including a "package" declaration at the top of each source file, classes can be organized into packages which results in several benefits. Classes in the same package have a special level of access to each other. Classes can be invisible to classes outside of their package, allowing implementation details to be hidden. The default of putting all classes into packages avoids the problems of global namespace pollution often encountered in large software projects. All of these benefits apply to *Sutherland* by making its collection of approximately 300 classes more manageable.

The standard libraries of Java also provide several advantages over C++ and other OOPLs. Important patterns such as Iterator (Enumeration) and Observer are provided as interfaces and abstract classes in the standard libraries, making framework implementation easier. Java also provides standard, portable APIs for GUIs, network access, and distributed computation. The `serializable` interface in Java 1.1 allows an object graph to be written to disk and then reconstructed with little extra programming. When *Sutherland* becomes more mature, each of these APIs will be useful in providing flexible and portable EC applications.

7 Conclusion

In this paper we've shown several examples of how the ideas of object-oriented frameworks and patterns were applied in the design of our EC framework *Sutherland*. In general we found these ideas to be extremely valuable, and we believe that *Sutherland* is significantly more flexible and extensible as a result.

We've used *Sutherland* in a variety of experiments (e.g., [McPhee *et al.* n.d.*a*]), and found it quite flexible. Use has, however, suggested a number of potential design improvements. Currently, for example, the various recombination operators (like mutation and crossover) are part of the `Rep_factory` subclasses. Our design would be more flexible if these were instead separate classes (examples of the Strategy pattern [Gamma *et al.* 1995]), which would allow us to mix and match collections of recombination operators without modifying the various factory classes.

Another planned extension of *Sutherland* is to implement the notion of a Visitor pattern for GP trees. The Visitor pattern [Gamma *et al.* 1995] represents an operation to be performed on the elements of a composite object structure. This pattern allows new operations to be defined without changing the objects they operate on. (in our case instances of the GP_tree_node class) This would allow us to have the GP tree classes represent *just* the structure of the parse trees without regard to other issues such as tree size, tree evaluation, etc. Visitors could then be defined to handle these other issues, and then attached to trees as necessary.

Ultimately, our goal is to make *Sutherland* available to the EC community in the hopes that they will find it useful. We also hope that *Sutherland's* flexibility and extensibility will allow other researchers to extend to *Sutherland's* functionality by adding new (sub)classes that implement different EC representations, recombination operators, and strategies.

Acknowledgments

Reierson's work was supported by the Morris Academic Partners program, and both Reierson and Hopper were supported by the Undergraduate Research Opportunities Program. A grant from the University of Minnesota Graduate School Grant-in-aid program program helped purchase computing equipment used in the development of *Sutherland*.

References

Gamma, Erich, Richard Helm, Ralph Johnson and John Vlissides (1995). *Design Patterns: Elements of Resuable Object-Oriented Software*. Addison-Wesley Longman.

Goldberg, David E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley.

Gosling, James, Bill Joy and Guy Steele (1996). *The Java Language Specification.*. Addison-Wesley Longman.

Holland, John H. (1962). Information processing in adaptive systems. In: *Information Processing in the Nervous System, Proceedings of the International Union of Physiological Sciences*.

Holland, John H. (1992). *Adaption in Natural and Artifical Systems*. MIT Press.

Koza, John R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

McPhee, Nicholas Freitag and Justin Darwin Miller (1997). Accurate replication in genetic programming. In: *1995 International Conference on Genetic Algorithms*.

McPhee, Nicholas Freitag, Nicholas J. Hopper and Mitchell L. Reierson (n.d.*a*). Impact of types on essentially typeless problems in GP. Submitted to GP-98.

McPhee, Nicholas Freitag, Nicholas J. Hopper and Mitchell L. Reierson (n.d.*b*). *Sutherland*: An extensible object-oriented framework for EC. Submitted to GP-98.

Mitchell, Melanie (1996). *An introduction to genetic algorithms*. MIT Press.

O'Reilly, Una-May and Franz Oppacher (1994). Program search with a hierarchical variable length representation: Genetic programming, simulated annealing and hill climbing. In: *Parallel Problem Solving From Nature - PPSN III* (Y. Davidor, H.-P. Schwefel and R. Männer, Eds.). Springer Verlag.

The Senior Research Project

Dr. Jean Mehta
Chair, Department of Mathematics and Computer Science
Saint Xavier University
3700 W. 103rd Street
Chicago, Illinois 60655
mehta@sxu.edu

Faculty members at small colleges often express a desire to give their undergraduate students the experience of a senior project (or research paper). It is widely believed to be beneficial to the students, enabling them to work independently, to explore a particular field of interest, produce a formal paper, and present the results to department faculty and seniors. While recognizing the advantages of *requiring* this as a capstone course for all seniors, we often hesitate to do so since it is difficult to implement, and can be an anxiety provoking experience for many students. It is especially troublesome for students at small liberal arts colleges where faculty members often do not have ongoing research projects in which students can participate. This paper describes a feasible procedure for incorporating a research requirement into the computer science major. Moreover, it is perfectly suited for the virtual classroom.

The Senior Research Project

Dr. Jean Mehta
Chair, Department of Mathematics and Computer Science
Saint Xavier University
3700 W. 103rd Street
Chicago, Illinois 60655
mehta@sxu.edu

Faculty members at small colleges often express a desire to give their undergraduate students the experience of a senior project (or research paper). It is widely believed to be beneficial to the students, enabling them to work independently, to explore in depth a particular field of interest, produce a formal paper, and present the results to department faculty and seniors. At job interviews, students can engage potential employers in a discussion of their project, thus demonstrating their maturity, self-motivation, and ability to work with little guidance. If this course is required of all graduating seniors, then faculty can also view it as an instrument of evaluation for the curriculum, a means of determining whether they have provided students with a firm foundation from which they can work independently to gather, evaluate, and present new material.

While recognizing the advantages of requiring this as a capstone course for all seniors, we often hesitate to do so since it is difficult to implement, and can be an anxiety provoking experience for many students. It is especially troublesome for students at small liberal arts colleges where faculty members often do not have ongoing research projects in which students can participate. These students are therefore required to suggest their own topic. They well realize the need to choose a topic early enough in the year so that sufficient time is available to devote to it, and yet they are uncertain how to choose one. Compounding this problem is the fear that they will not choose well, and that the project they undertake may be impossible to complete. Since this is also a time in their lives when they are trying to produce a resume and conduct a job search, it is very tempting to procrastinate on the senior project. This late start may result in a poor project or a failing grade and an inability to graduate.

At liberal arts colleges across the United States, faculty members have addressed these problems in their own unique ways ([Cunningham 1995], [Dawson-Howe 1996], [Neff 1997]). Those in the department of mathematics and computer science at Saint Xavier University have chosen a collaborative approach. This decision was reached after reviewing the literature on collaboration ([Johnson et al. 1991], [Treisman 1992]) and realizing that peer support and faculty mentoring would provide the students with the structure, encouragement, help, and guidance that they need. Moreover, within this collaborative setting, students choose their own research project. A further advantage is that with slight modification, this course is perfectly suited for the virtual classroom.

Methodology

Saint Xavier University has a joint Mathematics/Computer Science department with a total of ten full-time faculty serving approximately 50 mathematics majors and 60 computer science majors. Both programs (mathematics and computer science) include a required capstone course called Senior Seminar in which the student completes a research paper and presents the results to all department faculty and graduating seniors. (A computer science student may choose to complete a computer project; however, an accompanying paper is still required). Students must enroll in this course in both semesters of their final year.

During the first semester, the facilitator (usually the Associate Chair of the Department) holds weekly meetings with all students registered for this course and serves as their initial mentor. By exploring their fields of interest and suggesting periodicals and other resources (including faculty members) that may be of help, the facilitator works with them to choose a research topic. Group brainstorming sessions, at which students toss out ideas for projects and papers are also very useful since a topic suggested and rejected by one student may be exactly right for another student. If the facilitator is a mathematician, a computer scientist is invited to brainstorming sessions to help students develop ideas. (A mathematician is invited if the facilitator is a computer scientist). Since all students must meet the same deadline for providing a thesis statement, they encourage and support each other through this activity. By enforcing this deadline, students are not allowed to procrastinate in the choice of a topic.

The thesis statements are then reviewed by all faculty at a department meeting so that department members may state whether they feel a topic is appropriate, whether it appears to be too ambitious, or whether a slight modification would produce better results. This collaboration by department members ensures that the chosen topic is one that can succeed, thus relieving the student of the anxiety that the project or paper may be too ambitious. Individual mentors and second readers are also chosen for each student at this meeting. The mentor will be the primary contact person for the student, the second reader will be able to help the student if the mentor is unavailable, and will serve as the other faculty member who will read and grade the paper.

For the rest of the semester the facilitator continues the weekly meetings, helping students in their bibliographic search, and guiding them through the successful completion of an outline for their paper. The facilitator also prepares students for their oral presentation by demonstrating the use of various presentation media, and modeling good presentation techniques.

Each student-mentor pair meets and completes the outline of the paper as well as a timeline (which is a rough schedule for the research and writing of the paper). These outlines and timelines together with bibliographies are then handed in for discussion by department faculty during finals week of the first semester. At this meeting, each student's outline, timetable, and bibliography will be presented by the mentor. Students receive a pass/fail grade for this course as determined by consensus from department faculty. A failing grade results in the student having to repeat this first semester of Senior Seminar.

During the second semester the student works more closely with the mentor than the course facilitator. It is the mentor's task to help the student understand the material, to keep the student 'on track' according to the timeline, and to practice the oral presentation. All department faculty and graduating seniors attend these oral presentations and faculty members are asked to grade them according to certain guidelines.

Grading

At the end of the second semester all department faculty meet to discuss the final grade. 100 points are allocated as follows:

Mentor (for paper/project)	50
Second Reader (for paper/project)	25
Faculty Average (for presentation)	20
Facilitator (attendance at meetings and other students' presentations)	5

Mentors each bring a copy of their student's paper for review by other faculty if necessary. Mentors and second readers each state the number of points allocated for the paper and briefly explain why. The faculty average grade for the presentation is calculated, and the facilitator has five points to allocate for attendance. It is a simple task to then calculate a letter grade for each student. Moreover, there is consistency in grading.

Results

Few students have been unable to successfully complete the first semester. In most cases this failure was due to family crises or other outside stresses. After repeating this first semester they finished the second semester without further problems. Rarely has a student withdrawn from (or failed) the second semester.

It is advantageous to the student to have the mentor's help in setting up the timeline. Often the student has unrealistic expectations, and wishes to create a timeline that is impossible to keep. However, the mentor will help to create one that is practical.

Some students seek more help than others from their mentor. Nevertheless, most mentors see their students on a regular basis in order to help them meet the deadlines they imposed on themselves in the timeline. Personally, I like to see my students for about an hour every two weeks.

Mentors report that they develop strong bonds with their students in working together with them, albeit in an advisory capacity, on their project or paper. Faculty also enjoy working with other members of the department to guide the seniors in their project.

Most recently, we made a decision to have all presentations on Saturdays to enable family and friends to attend. Students have been appreciative of this, and parents have enjoyed meeting the faculty and being present at their son's or daughter's senior seminar.

Some of the most recent topics in the computer science Senior Seminar include:

- Neural Networks
- Real-Time 3-D Graphics Imaging
- Intelligence Agents
- Object Oriented Databases
- Data Compression
- Web-based Instruction
- World Wide Web Database
- Programming with Direct-X

We have found that this process gives the students the structure that they need. Since the mentors help them to stay on track, it is very difficult for them to procrastinate. They know that many faculty members have looked at their proposal, outline, and timeline and so they have confidence that this is a project that is possible in the allotted time. Furthermore, they have a mentor to guide them through the process, suggest resources and explain concepts to them, if required. Finally, in the words of a recent graduate, "I hated Senior Seminar, but in my new job that's all I seem to do – life is one huge Senior Seminar! Thanks for showing me how to write and present my ideas."

Conclusion

I have described a feasible procedure for incorporating a research requirement into the computer science major. Moreover, it is perfectly suited for the virtual classroom. In the first semester a web page may be established for the course and will serve as the primary means of communication. The brainstorming sessions can be conducted using electronic discussion groups. The facilitator can also communicate via e-mail with all students enrolled in the course.

It is undoubtedly time consuming for faculty members who receive no compensation for this (apart from the course facilitator who receives credit for one course). However, we believe that it is a valuable component of the curriculum. It provides us with a means of evaluating our program while giving the students the experience of working independently to produce a research paper (or computer project) that will be presented formally to parents, seniors, and faculty.

References

[Cunningham 1995] Cunningham, S. J., (1995). An Introduction to Research and the CS/IS Professional Literature for Undergraduates. SIGCSE Bulletin, 27 (4), 5-8.

[Dawson-Howe 1996] Dawson-Howe, K. M., (1996). Organization of Final Year Projects. SIGCSE Bulletin, 28(3), 2-4.

[Johnson et al 1991] Johnson, D. W., Johnson, R.T., and Smith, K.A., (1991). Active Learning: Cooperation in the College Classroom. Edina, Minnesota, Interaction Book Co.

[Neff, 1997] Neff, N., (1997). Research Seminar: The World Wide Web on the World Wide Web. Journal of Computing in Small Colleges, 12(5), 106-112.

[Treisman, 1992] Treisman, U., (1992). Studying Students Studying Calculus: A Look at the Lives of Minority Mathematics Students in College. College Mathematics Journal, 23(5), 362-372.

Security and Flexibility: Computer Labs in a Small College

Kevin Olson
Senior, Management Information Systems
North Dakota State University – Student
kevolson@plains.nodak.edu

This presentation will describe the methodologies used to implement and maintain certain sets of software in the Windows 95 computer labs at North Dakota State University. Due to the teaching environment there needed to be a way of providing a flexible software configuration that allowed for the teaching of the operating system, while at the same time maintaining consistency for the drop in users. The need was also there to remotely install and configure the software on the computers without actually going to each individual computer and manually installing it. The methodologies that have been developed center around a set of small utilities that make the technical professionals more manageable while at the same time expanding the set of services found in the clusters.

Security and Flexibility: Computer Labs in a Small College

Kevin Olson
Information Technology Services
North Dakota State University
kevolson@plains.nodak.edu

The ideas of security and flexibility can, of course, mean a number of different things depending on the environment where the terms are used. For the purposes of this discourse however the terms will be defined within the context of a specific audience, namely that of the users of cluster (public lab) computers on a college campus. Security can be defined in this context as measures taken to protect the software on the computers. More specifically, these measures are taken to guard against both accidental and malicious manipulation of any software that is found on the machines. Flexibility, within this context, can be defined by the ability of the methods used to maintain the cluster software to adapt to the changing requirements that are demanded by users. When the idea of flexibility is discussed it is almost always within the framework of adding value to the existing software environment. Fulfilling the specific needs of different groups of users brings about this increase in value. As the discussion of the above issues develops it becomes increasingly important to realize that they are not mutually exclusive, a balance can be reached. Furthermore, the issues that concern security are not, by any means, contradictory to the goals that flexibility would like to fulfill; rather, in many instances, they compliment each other quite well. The purpose of this paper, then, is to take a step back and ask some general questions about public labs. What are these labs for? What forces drive their acceptance and usage? What challenges are presented to the technical staff? The discussion will also deal with answering these questions and what those answers mean from an implementation perspective for the internal support function at small colleges. Lastly the technical details of the methods that are used at North Dakota State University will be presented. The technical details will be at the center of how the balance between security and flexibility is being achieved and will be presented with that in mind. With respect to context, it should also be said that these technical details will include the way that the Intel-based machines running Windows 95 are handled.

Macro View of Public Clusters at NSDU

To begin to further understand the issues involved in implementing sound solutions to the situations that arise it is helpful to get an idea of just what NDSU has in terms of resources (number of clusters, number of machines) to provide to the users. Currently there are 14 rooms that have approximately 220 PC compatible computers along with a portable cluster that consists of 14 laptops that are used in various locations. There are also 9 rooms with approximately 140 Macintosh compatible computers, but for the purposes of this paper the scope will be limited to the discussion of only the PC compatible computers. Of the 14 rooms that contain PC compatibles 10 of these are available for reservation by faculty, staff or student use as long as the purpose is educational in nature. The hours that the rooms are open varies based upon the building that the room is located in and usage statistics. There are currently 3 rooms that are open 24 hours a day with most of the rest of the rooms following roughly a 6 a.m. to 12 p.m. format. The purpose for mentioning these facts is that the high amount of time that the clusters are available can cause significant problems when it comes to testing out new ideas or even implementing needed features. The window of opportunity for changes and reconfiguration is narrow for most rooms and simply non-existent in others. The matter of availability is among the first issue that is faced when considering how to approach the technical details involved in creating a complete, coherent system to maintain the software on the cluster computers.

Practical Uses of Clusters

As the macro view of cluster technology on a given campus continues the issue of what exactly the clusters will be used for arises. This issue also lends itself to a better understanding of who will be using the clusters. It would be inefficient, if not impossible, to list every potential scenario that would represent a purpose that the clusters would be used for but some examples could help narrow the focus a bit. The following are examples of situations that have been occurred here at NDSU and are most likely fairly common scenarios throughout most of higher education. There is, of course, one of the most noticeable uses that crosses practically every user that comes into the cluster, electronic mail. There are also many classes that teach software packages right in the clusters as well as the internal information technology staff teaching the various campus specific software packages to faculty and staff in the clusters. External organizations are sometimes called upon to teach or present something to students, faculty or staff and would like to use the multimedia resources that are available in the clusters. There are, of course, many other different examples that could be cited as uses for the public clusters. Just some statistics on the number of executable programs that can be launched from any given cluster machine alone can be overwhelming. There are close to 1000 executable programs on these machines with over 2500 different support libraries for the programs. Consider the following example for an idea of just how the practical use of public clusters can become a difficult job for those involved in maintaining them. Let's say that you have Microsoft Office 97 installed for access from the cluster machines. Further consider that this installation is set up to run from a particular server due to the size constraints of the local hard drivers and also for ease of administration. Let us also say that many other programs, like Visual Basic 5.0 for example, are installed and may have to run simultaneously. Not only are these products from the same company but they shared a certain amount of functionality through the use of shared-code libraries. Now, lets say that the situation arises where those who do training on applications would like to teach staff members the intricacies of a previous version of one of the programs that are currently installed on the cluster machines, for example, Office 95. The reasons for such a request coming in are quite valid due to the fact that many of the offices on the campus at NDSU still use the older versions of some software. The reasons that they still use the software can be many but usually include the level of comfort that they may feel with the software, as well as the cost of upgrading to a new version of the software. The idea of cost, in this context, may involve not only the dollar amount that a department may have to pay out for a new version but also the intangible costs of having everyone in the department switch to something new. All of this leaves the cluster technical staff, or those responsible for getting this all to work, in quite an interesting position and with a number of different options. It may be against cluster policy to replace the newest version of a particular piece of software with one that is older. It would almost certainly create havoc amongst the everyday users of these pieces of software, who would find forwards compatibility problems with the output that they might have created with the latest version. What about the programs, like the aforementioned Visual Basic, that rely on some consistent set of shared-code libraries and registry entry keys to get things done within the program?

Forces that Drive Acceptance and Usage

Now that the users of the cluster machines have been identified and an example of what can occur within the scope of practical use has been put forth more detailed questions can be asked. What are some forces that drive the acceptance and usage of the cluster computers? Why do faculty, staff and students continue to increase their usage of these resources? These questions have become increasingly more relevant in this age of higher processing power to be found for lower prices. Many students, it would seem, should be using their own computers to do some of the work that they have to do. The same idea should hold true for faculty and staff with the high amounts of resources that are being funneled to increasing the technology infrastructure. The definition of force, within this particular context denotes some type of strength or energy that has brought about a particular movement. That movement in the clusters has been increased use in the past couple of years along with an increasing interest among faculty and staff to expand the services that are offered by the clusters to other parts of campus. The forces are indicative of a certain set of properties that surround the clusters that are benchmarks for the performance of the clusters in the minds of those that it serves. Moving along to the word acceptance it is plain to see that it is perhaps one of the more difficult terms to really define in this context. However, it would seem that acceptance can be classified as an attitude that the users of the clusters possess about the resources that are available for use in the clusters. Perhaps the issue of acceptance could be further clarified by way of example. Let's say that a common piece of software is used for an introductory class and one of the projects involves performing a task that simply cannot be done due to some incompatibility. Now, the reason that this task cannot be done does not lie with something that could be fixed at the moment, for example a program anomaly that has yet to be resolved by the manufacturer. However, no one involved with the particular project, either the teacher, student or cluster technicians were aware of the incompatibility thus leaving all unprepared. The way that incidents like this affect all parties involved can be directly related to their level of acceptance of the cluster technologies. It is in this vein that the word acceptance will be used and applied from here. If a person were to have some level of acceptance of cluster technologies then it would follow that they would have to have some task to perform in the clusters. The issue of the growth of both electronic mail and the World-Wide-Web has certainly fueled the interest in making public clusters more available to students, faculty and staff. Now, this interest alone does not necessarily mean that the clusters will be used or accepted. If the infrastructure, as it currently exists, did not allow for a reasonable experience with these new technologies then the demand would certainly not be as high for the services of the clusters. That is to say that usage and acceptance of the clusters on a given campus does not depend solely on the correct implementation of the means to take advantage of new technologies. The correct implementation of these technologies, is however, definitely one of the important factors that would drive the acceptance of the clusters. The reason that the correct implementation is so important is because it is by far the most readily seen component in the process and has been, historically, the easiest piece in the puzzle to place the blame on. This means that, the technical professional in charge of the software distribution on the cluster machines must make sure that the tools that make all of this technology available works well every time. Another factor that can drive acceptance is the amount of comfort that any given user feels when sitting down to use one of the computers on campus. It is human nature to want to be able to be familiar with a certain interface; it increases our level of control over our environment, which, in turn, makes us more comfortable with whatever it is we are dealing with. One giant step in acceptance of cluster technology, at NDSU anyway, has been a consistent desktop across all machines. This consistent desktop would include, but not be limited to, a common group of desktop icons, a common screen saver and a common set of icons and groups of icons in the Start Menu. It is technical details like these that enhance the forces that are at work to drive acceptance thereby, increasing the overall use of the clusters.

Challenges Presented

There are many challenges that the information technology function as a whole will face with regard to situations like the one's described here. A properly built networking infrastructure, a well-managed team of technical staff and a good working relationship with all of the users are important challenges that must be faced. From an implementation perspective there are a great many challenges as well. Given the previous examples there must be a way of maintaining uniformity across all of the desktops and distributing software to all of them as well. It also would be good to have methodologies in place that would compliment environments where the actual hardware that is present would not matter as much as it has in the past. Other challenges that will be found are how to protect the software on the machine so that it remains consistent every time a user sits down. Along with all of these challenges lies a fundamental question about the way these things can be accomplished. Can the technical professional meet all of the challenges presented here with methods that are efficient and effective?

The Foundation Laid

The discussion thus far has focused on some practical examples of what the environment is like on a small college campus with regard to these cluster issues. Taking an overall look at what the purposes are for the clusters, what forces drive their acceptance and what challenges will be faced along the way is certainly a worthwhile place to start. By taking a honest look at the clusters from an overall standpoint the technical professional begins to see not only their own point of view but also how the cluster is viewed from the perspective of other peers upon whose expertise the success of the clusters also rests. It quickly becomes clear that in order for any public cluster to satisfy the needs of the greater college community an information technology department must meet the challenges that will be faced upon the way as a coherent whole. It is an absolute necessity that all persons involved in the activity that takes place in the clusters feel free to exchange ideas with the knowledge that their input will be taken seriously. Everyone from the person that refills the toner cartridges to the individual who writes out the budget must be able to feel comfortable to contribute to making the public clusters a place of constant improvement. Now as an expected result of looking at the overall picture first we know that eventually the time comes to move into the particulars involved. There are, of course, many particular tasks that must be accomplished to run a cluster, especially when taking into consideration all the details that are involved on the maintenance end. The questions that were raised earlier are, in and of themselves, enough to cause the realization that the matters of security and flexibility are ones that must be addressed. The particular details involved in answering the questions, with security and flexibility in mind, will be the topic for the remainder of this paper. The focus now will shift from that of a high level, overall understanding of how the clusters function as whole and what technical challenges must be overcome to one that will define how exactly to overcome these challenges in a way that compliments the overall function of the clusters.

The Technical Process

Now that the groundwork has been laid for a proper understanding of what is needed in the clusters, from a technical perspective, one has to consider just how this will be done. The remainder of this paper will, as was mentioned before, serve the purpose of describing just how this is done at NDSU. The beginning part of the discussion will focus on the software tools that are used to help accomplish this task. After this the focus will move into some of the more subtle methodologies that are used as well as some of the special procedures that occur that make the job easier or better. One other method that will be used is to try to document the process of making a machine suitable for cluster use, sort of watch it grow from the time you take it out of the box to the time you let it go into the world of the students, faculty and staff. Lastly it would be helpful to describe a "day in the life" of a cluster machine, in terms of how things happen to the machine and how the restore procedure at night works. This discussion will also include how all of the software interacts to make the restore procedure a possibility.

Cluster Support Software

PC-Rdist

PC-Rdist is at the center of what happens with respect to maintaining and implementing all software on the machines in the clusters. In brief, PC-Rdist is used here to provide a reasonable means for keeping the hard drives on the individual computers refreshed without having to worry about where they are at, what printers they print to, or even what hardware they have. It provides us the flexibility to perform everything from the base level formatting of the hard drives and restore of the machines to small things like an overnight subtle change such as a new bookmark in Netscape. The person who wrote this software goes by the name of Chris Volkert. Chris has started his own company based on the popularity of his PC-Rdist product; the company is called Pyzzo Software and detailed information about his company can by found at www.pyzzo.com. Chris has offered quite a bit of personal help out to the users of PC-Rdist through a mailing list that can be subscribed to. Instructions on how to subscribe to the list can be found at the web site. Currently at NDSU we are running version 1.55 of PC-Rdist and have found it to be quite stable in all respects. The many functions that PC-Rdist performs are too numerous for the purposes of this paper but the following will be a basic outline of what functions have found to be most appropriate for our tasks. PC-Rdist comes in two different executable forms, one distribution runs under both Windows 95 and Windows NT, and the other distribution runs under DOS. At NDSU the DOS version of PC-Rdist is used in order to start a machine out from scratch, to get just Windows 95 and the Novell Client software on it. After Windows 95 is installed and can connect to the network then the 32-bit version of PC-Rdist takes over and does the rest of the work. Both versions of PC-Rdist must be launched from the command line and can take the following parameter list when being run. Refer to [Tab. 1] for an explanation of the meaning of the parameters that can be used when running PC-Rdist.

```
pcrdist [-ncpilvqx] distfile
```

Parameter	**Meaning**
-n	Print commands without executing them. This is useful for debugging distfiles.
-I	Interactive Mode. PC-Rdist will pause after each section of the distfile and let you examine the master image, local drive, temp directory and distfile. (Win32 only)

-p	Prompt before executing any commands.
-l	Show the log window by default. Normally it is hidden unless you press ALT-L. (Win32 only)
-c	Print the distfile to the screen after it is preprocessed.
-v	Verbose mode. Prints commands as PC-Rdist executes them. Useful for debugging distfiles. (DOS only)
-q	Quiet mode. Don't show the main window.
-x	Exclusive mode. Make sure there isn't another copy of PC-Rdist running.
Distfile	The file to use for configuration information.

Table 1: PC-Rdist parameter meanings [1]

The parameters that have been found to be most helpful here when installing and configuring software have been the –n, -i and –l parameters. These will be looked at a bit more in depth during the discussion of how to actually go about installing a particular piece of software. The distfile is an important part of the process due to the fact that it is within the distfile that PC-Rdist looks for the rules that will define just what will go onto a particular hard drive. The distfiles composition is just plain text that can be viewed by any regular ASCII text editor, it is important to note too that the file must be readable by the user that will launch PC-Rdist if the distribution files are going to be stored on a network drive. The distfile can contain a number of configuration variables that PC-Rdist will recognize; these variables define how the program will handle certain situations. The distfile also contains a section that is called the distribution block. Within the distribution block one can provide a description of what the local hard drive is to look like. The description of the local drive can be in terms of the directory structure and the internal structure of the registry.

Getha

Getha is a small DOS based utility that is used to set environment variables based on the hardware addresses of the ethernet cards or the serial numbers off of hard drives. Getha can also be used to obtain the hardware address of a particular ethernet card or even the serial number of a hard drive. Hardware addresses are unique, twelve digit (hexadecimal) identifiers that are part of an ethernet card. In much the same way as hardware addresses are unique to ethernet cards, serial numbers are unique to hard drives; however, there are many programs that can be used to change the serial number on a hard drive which makes it an inherent security risk. Getha can be used under both DOS and any 32-bit Windows platform with much the same functionality. Environment variables are operating system level identifiers that remain resident in the parent process where they were set. Consider this example, the DOS path command can be set as an environment variable and used to tell the operating system where to look when an executable name is typed on the command line. If the path command is set in the file autoexec.bat you can open up a DOS window and type the command ‘set’ and find out what the path command is equal to; conceptually then, Getha works in the same way. It is important to remember under the 32-bit platforms that if you run Getha the environment variables will only be set for the parent process. The parent process for Getha will be the DOS prompt where are you executing it from, so if you launch another DOS prompt you will spawn another process, thereby creating a new environment in which those variables will not be set. Getha will accept the following command line, followed by an explanation [Tab. 2] of the parameter meanings.

```
getha [-s -v] [-t token_seperator] -f dbfile
```

Parameter	**Meaning**
-s	Use hard drive serial number instead of ethernet address
-v	Verbose mode: print values only, don't change environment
-t	Use token_seperator to separate variables in the db file. The default token separator is ":" (colon)
-f	Use data file filename

Table 2: Getha parameter meanings [2]

[1], [2] taken from www.pyzzo.com

The main purpose for using getha here at NDSU is to allow for higher flexibility within the distribution files. PC-Rdist recognizes the environment variables that are set within its parent process and understands how to use them within the distfile. The dbfile that getha references is an ASCII text file and has the following form contained on the first line, followed by a two-line excerpt from the getha dbfile that is used at NDSU.

```
HA=XXXXXXXXXXXX:VARIABLE1=VALUE1:VARIABLE2=VALUE2:...VARIABLEN=VALUEN
HA=00c04fd7cb19:NAME=cq01:ROOM=CQ:TYPE=DELL100
HA=00c04fd7ca69:NAME=cq02:ROOM=CQ:TYPE=DELL100
```

The first thing that is listed is the hardware address (HA) that identifies the machine followed by all of the environment variables (VARIABLEN) and corresponding values (VALUEN). So, for example, if getha were run on a file containing the excerpt from the dbfile found above and had the hardware address 00c04fd7ca69 the Name, Room and Type values would be set. Then, within that parent process, the set command would show the following associations; NAME=cq02, ROOM=CQ, TYPE=DELL100. PC-Rdist could then use these variables by inserting their values where the variable names were referenced in the distfile. One of the main reasons that this is significant at NDSU is that it allows us to have one master distfile that transfers control to a room specific and machine specific distfile based on the environment variable values found in the Getha dbfile.

Regdump

By far one of the most intimidating aspects of distributing Windows 95 across many different types of machines with a multitude of hardware configurations is this matter of the registry. Regdump is a utility that makes the dark cloud that is the registry a little bit easier to deal with. The Windows 95 registry is basically a centralized location for system information, like hardware profiles and version numbers, and application-specific settings such as bookmarks in Netscape. This information is stored in what are known as keys, which themselves can contain values. One of the well-known reasons for why the registry exists is to replace many of the problems associated with having multiple ini files. These problems ranged from a lack of standards all the way to the time it took programs to look up information within these files. Unlike the ini files however, the registry is not accessible for editing in the same way as ini files are. An ini file is just a plain text file so if you want to change something about a particular entry all you have to do is open up the file in a text editor and change the file. The registry, on the other hand, is only accessible through an application-programming interface (API) provided by Microsoft that will allow the binary information in the registry to be exported to plain text. In the same way the API will allow plain text files to be imported into the registry as well. Regdump itself uses the API to write the data that is found in the registry files into text files. Regdump can be used with the –o parameter in the following way.

```
regdump -o somefile.reg
```

What this will do is place the all of the data that is found in the registry into the file somefile.reg. The other feature of Regdump is the ability to compare a before and after snapshot of the registry and give the user a file containing the difference. Consider this example; let’s say you would like to find out where the data concerning the screen resolution is stored. In order to find out this information you would run Regdump, then make the change in the screen resolution and run the following command.

```
regdump before.reg –o resolution.reg
```

The file before.reg is the existing registry data that came as a result of the initial Regdump. The file resolution.reg is where the difference between the before.reg registry and the current (newly changed) registry will be stored. The power of these registry dumps becomes apparent when taken into consideration with the fact that the distfiles that PC-Rdist reads from can contain references to these files. A more detailed discussion of how this is applicable will be covered in the section regarding software installation and configuration.

Policy Editor

The policy editor is 32-bit Windows application from Microsoft that takes advantage of the Windows API to make direct changes to the registry through a graphical user interface. It is with this tool that the disabling of some of the functionality of the Windows 95 interface has occurred. This program can be found on any Windows 95 CD-ROM in the /admin/apptools/poledit directory. The name of the executable program is poledit.exe. The policy editor can be used to set up environments for individuals users based on the Windows login name. One of the environmental things that you can change with the policy editor is access to certain portions of the Windows operating system. With only a few exceptions every machine on NDSU's campus has the same amount of restriction when it comes to accessing certain portions of the operating system. The policy editor is used in conjunction with Regdump to find out what changes in the registry occur when limiting access to all of the control panels as well as removing the Settings folder off of the Start Menu. It is at this point then that the changes to the registry keys can be placed in a file and referenced by PC-Rdist in order to become part of the nightly restore.

InstantON

Many times during the course of considering the best ways to go about maintaining cluster software the problem comes up that the majority of the clusters are open for an extended period of time. This creates some difficulty in actually finding time to maintain and upgrade software on the computers. Even if the actual process itself is fairly automatic there still needs to be some sort of user interaction to start the update. It is at this point that a small freeware utility from Intel called InstantON comes into the picture. In brief, InstantON keeps track of the system time on the individual computer and can be configured to launch applications at a specific time and date. The way that this is used at NDSU is simply to start a batch file early in the morning that begins the restore process. This batch file is run every day between the hours of 4 a.m. and 5 a.m. and is stored on the local hard drives. This way changes that need to take place overnight are assured of happening without any intervention by the technical staff. Even if there are no changes that need to be added this process will at least remove any unwanted or unnecessary files and return the machines to the proper configuration.

A Day in the Life

Now that the individual pieces of software that are used to help support that cluster restores are defined lets take a look at a typical day for a cluster computer on NDSU's campus. Typical, in this context, can be defined in terms of what should happen on a day to day basis in terms of the restore procedure as well as an how the initial setup of the machine should take place. It is within this discussion that the finer details of how all the software interacts will become clearer. Examples of some special situations that occur at NDSU will be covered along with an overview of how they are handled.

Right Out of the Box

Usually when machines are ordered for general cluster purposes they are ordered in a fairly large quantity. The advantage of having a restore procedure is that, by setting one machine up, you can have all of the other machines copy what is on the first machine. The following may be rather elementary to many of the readers of this; however, this section will hopefully lay a solid groundwork from which to build a decent restore procedure. I'll try to be brief where it seems appropriate. To begin with, I'll take one of the machines out of the group and get everything all set up. Then, making sure that I have all of the necessary drivers for all of the internal and external devices I'll continue by formatting all of the drives that are in the computer. Windows 95a is run exclusively here at NDSU so I would continue by installing Windows 95a with only the options that are needed in the clusters. It is at this point I would set up any of the little things about the operating system that need to be set. Some of the little things that can be done include turning on Auto Arrange for the Desktop icons and turning on Show All Files in Explorer; this will save quite a bit of work later on and you will be glad you did it. Currently at NDSU file and print services are being obtained from a NetWare server running IntraNetWare 4.11 so I would proceed by installing the latest client software and configuring for use within the cluster. Note that at this point there is no defining of any users that will login to the server or any printers that they will print to, just a basic client install that will allow anyone to use the machine to login. One thing to note as well is that it is here that I disable the login prompt for NetWare password. What happens in the clusters is that the student automatically is logged into the NetWare server due to a login shortcut in the Startup menu. Once all of the initial prepping is done to everything on the hard drive I'll proceed by copying the contents of the entire directory structure up to a location on the server. If, for example, the machines happened to be Dell Pentium 100's I would copy the contents to a directory like 'f:\restore\dell100'. It is critical at this point to understand that everything must be copied up to the server, including hidden, system and read-only files. This can be accomplished through the XCOPY command that this included with Windows 95. To learn more about XCOPY simply open up a DOS prompt and type 'xcopy /?'. With all of the files up on the server I would then create a small distfile that the DOS version of PC-Rdist could read from in order to place these files on all of the other computers that need this software. The distfile that controls this portion of the restore would look something like the following.

```
#DEFINE DELL100DIR F:\restore\dell100
tempdir=C:\
> c:\ %DELL100DIR% : m s d t r
<
```

You will notice that this distfile follows the convention that was set forth in the section that dealt with PC-Rdist. There is a clearly defined configuration variable section in the first two lines and then the distribution portion following that. In plain English the distfile would read like this; after you start PC-Rdist set up a defined variable for the duration of the processing of this script called DELL100DIR. The variable DELL100DIR is to stand for the directory F:\restore\dell100, also tell PC-Rdist to store temporary files within the root of the C drive. Then PC-Rdist will place the files found in DELL100DIR onto the local hard drive according to the rules that have been set up. The directory structure and registry structure of any local hard drive can be updated based on a certain set of rules. Within this distfile the files with be replaced if it does not exist on the local machine (m). The file(s) will be replaced if the size has changed (s). Files will be deleted (d) if they exist only on the client machines. Files will be replaced if the timestamp (t) is different of a file that exists both on the client and the server. Finally, these rules will apply to all subdirectories of the root of drive C (r). All of these rules, as they relate to Windows 95 and DOS, are found in [Tab. 3] as follows.

Flag	Mnemonic	Meaning
M	Missing	Replace the file/key if it doesn't exist on the local machine
S	Size	Replace if the size has changed
T	Time	(filesystem only) Replace if the time has changed (short for O N)
O	Older	(filesystem only) Replace if local copy is older
N	Newer	(filesystem only) Replace if local copy is newer
A	Attributes	(filesystem only) Reset file attributes if they have changed
D	Delete	Delete if exists on the local machine only
J	Junk	(filesystem only) Move the file to the temp directory if it exists on the local drive only
I	Ignore	Ignore this file/directory/key on both the client and the server (short for IS IC)
IS	Ignore on Server	Ignore on the server
IC	Ignore on Client	Ignore on the client
R	Recurse	Apply this line to files in subdirectories as well
B	Reboot	Restart the computer if this file/directory/key is replaced
W	Wait to Replace	(Win32 only) Replace this file/key when the computer is restarted
C	Checksum	Compare checksums for the master and local copies
CASE	Synch Case	Synchronize the case of master and local filenames

Table 3: PC-Rdist Action Flags [3]

[3] taken from www.pyzzo.com

All that is really needed after this point in order to get the rest of the machines up and running is a boot disk. This boot disk must contain the drivers that will allow the machine to connect to the network under DOS. What happens then is very similar to what would happen on a fully functional Windows 95 machine, a user would login in automatically, Getha would be run to set up certain environment variables, then the DOS version of PC-Rdist would take care of getting all of the files onto the hard drive. Note here that an inherent restriction of copying Windows 95 down to a hard drive under DOS is that the long filenames that are contained in certain portions of the Windows directory structure (i.e. Start Menu) will be truncated.

Growing Up Fast

Now that a reasonable means for getting Windows 95 and the network client on the local hard drives is established it is time for the rest of the software to be installed on the machines. It has always seemed like software installation, and more importantly the conflicts that can arise afterward, have been such a nebulous and misunderstood concept where most people, including technical professionals, really do not know what is going on. When it comes right down to it, though, there really are a small number of things that can happen when a piece of software is installed. During installation files can be added to the hard drive and existing files can be modified, which includes additions and changes to the system registry, it is that simple. Now, the complexity part come when trying to manage the 1000 different executables and 2500 support libraries and finding the conflicts there, that is why good documentation of just what a piece of software installs is vital. So lets say that we have this new machine all ready to go and we would like to install the latest version of Perl as the first piece of software. The first thing that would happen is I would run the following command, assuming that Regdump is stored locally on the hard drive.

```
regdump –o before.reg
```

This would take a snapshot of the existing registry, as explained earlier, and store it in the file before.reg. Now, because this is the first time that Regdump has been run on this particular type of machine it will be used as the base upon which all other registry files will be added. What happens then is that I would rename the file to something meaningful like dell100.reg (also leaving before.reg intact so it can be used later) and copy it to a central location on the file server. Then I would proceed to install the Perl software distribution just like any other piece of software, placing it either locally or on the server. After the installation I would reboot the machine and then run Regdump a second time to see if this installation added any keys to the registry. The command line and contents of perl306.reg would be as follows.

```
regdump before.reg –o perl306.reg
```

```
REGEDIT4
[HKEY_USERS\.Default\Comments]
"perl306.reg"="#"
[HKEY_LOCAL_MACHINE\SOFTWARE\ActiveWare]
[HKEY_LOCAL_MACHINE\SOFTWARE\ActiveWare\Perl5]
"BIN"="C:\\Perl\\bin"
"PRIVLIB"="C:\\Perl\\lib"
"HTML-DOCS"="C:\\Perl\\docs"
```

As can be seen here the software did in fact make some registry entries that will need to be added to the distribution. Then I would copy the file ‘perl306.reg’ to the fileserver in a central location where all of the registry files will be stored, for example, f:\restore\regfiles. The central location is important for two reason, one, PC-Rdist looks for registry files in one directory and two, it is nice to have one location for all of the registry files for management purposes. Now that the registry information is stored and safely tucked away on the file server lets try to find out exactly what files were stored on the hard drive when Perl was installed. What we will need to do is launch PC-Rdist with the appropriate distfile for a Pentium 100. By making a copy of the distfile that was used to restore the machine from scratch a new distfile can be built that will have base for the software that will be installed. Our original distfile looked like the following.

```
#DEFINE DELL100DIR F:\restore\dell100
tempdir=C:\
> c:\ %DELL100DIR% : m s d t r
<
```

Let’s modify that slightly now to contain the following lines

```
#DEFINE DELL100DIR F:\restore\dell100
#DEFINE SoftwareDir F:\restore\software
regpath=f:\restore\regfiles
tempdir=c:\windows\temp

> registry dell100.reg perl306.reg : m s c d
<
> c:\ %DELL100DIR% %SoftwareDir% : m s d t r
<
```

The modified version of our original distfile should look something like what is seen above. The file should be given a unique name that will identify the unique properties that will make up the environment that the machine is in. For example, at NDSU if there were Dell Pentium 100’s in room 150 of our computer center I would name the distfile for that particular machine d100ch.dst, where ‘ch’ is the two-letter room identifier that is used at NDSU for room 150. As you can see there are a few new things in the modified distfile. The item that may cause a few raised eyebrows is the ‘regpath’ directive as well as the registry distribution block. The regpath directive just tells PC-Rdist where to look for the registry files that will be referenced in the upcoming registry distribution block. Within the registry distribution block PC-Rdist takes all of the files found there, combines them into one large file and creates new binary registry files from them. This is one of the most powerful aspects of PC-Rdist. The ability to take arbitrary text files that contain information that can be stored into the registry and import them on demand makes for a very flexible means of installing the same software on many different types of machines. The reason that this is so flexible is that the person installing the software can find out exactly what any given piece of software does to a machine without regards to the hardware thereby making the distribution of this software to other types of machines relatively simple. Now, getting back to finding out just what files were installed with this piece of software PC-Rdist has to be run again with the modified distfile in place.

```
pcrdist –n –i –l d100ch.dst
```

This will launch PC-Rdist in interactive mode, which will just show the user what would have happened to the local drive with the given set of rules. The –l switch will simply bring up a log window to give an account of what actually would have happened. The contents of this log file look like the following

```
...
Scanning c:\ ...
4011 files scanned
Scanning temp directory ...
0 files scanned
Tempdir is 0 bytes
Deleting file c:\Perl\bin\Cmd32.exe (delete flag)
Deleting file c:\Perl\bin\Perl.exe (delete flag)
Deleting file c:\Perl\bin\Perl300.dll (delete flag)
Deleting file c:\Perl\bin\PerlGlob.exe (delete flag)
Deleting file c:\Perl\bin\perlw32-install.bat (delete flag)
Deleting file c:\Perl\bin\perlw32-install.txt (delete flag)
Deleting file c:\Perl\bin\perlw32-uninstall.bat (delete flag)
Deleting file c:\Perl\bin\install.log (delete flag)
Removing directory c:\Perl\bin
```

```
Deleting file c:\Perl\docs\Perl\license.txt (delete flag)
….
```

There is quite a bit more to these log files but this gives the basic idea of what the important parts look like. Based on the rules that have been set forth in the distfile PC-Rdist will report what actions should be taken with regard to any file that is affected by these rules. For example, you will notice in the log window output that PC-Rdist would have deleted the files within the Perl directory, this is because the files contained therein are not yet part of the directory structure that is identified in d100ch.dst. What happens next is that the log file is gone through to find out just what files need to be copied over to the file server in order for the software (Perl in this case) to be installed on the local drives after PC-Rdist is run.

Step by Step Overview

To sum up some the discussion of installation of software and the use of these small programs to the cluster hard drives let's take a complete look at all of the things that happen (in order) to a machine at night when it restores.

1. If the machine is not on ten minutes before it is supposed to restore the bios is set to turn it on automatically. This step simply cannot be accomplished on those machines whose bios does not support automatically turning a machine on.

2. The machine is then idle until the time that InstantON run the batch file to clean the machine (clean.bat).

3. The clean batch file then does a check to make sure that the computer is properly logged in to the network. If the computer is not logged into the network the batch file goes through the process of logging in.

4. The clean batch file then launches Getha and sets the appropriate environment variables.

5. PC-Rdist is then run with a master distfile which, based on the variables set with Getha, will transfer control to the room/computer specific distfile that should be parsed.

6. PC-Rdist then performs actions based upon the rules set forth in the distfile and completes the task by sending email to a specified address notifying them of a successful completion for that machine, and then reboots the computer.

All of these steps should seem familiar given the previous discussions with the possible exception of the email message in number six. That message is an important part of maintaining the clusters here at NDSU. By PC-Rdist generating an email message with a particular exit value we can pinpoint just exactly what machines did and did not restore based upon the unique identifiers of the machine. Basically what happens is that the mail gets sent to a specific Unix account and is automatically saved to a file. Then, after all of the cluster computers restore at night, there is a mechanism that automatically starts processing the mail file that has been building up. The result of the processing is a list that is posted to a web page that tells the hardware technicians which machines did not restore that night.

The Future

The questions that were asked during the course of this paper dealt with some of the overall issues that are found within cluster environments on small college campuses. The answers to those questions allowed us to begin to focus on the technical side of a cluster implementation and examine just how this is done at North Dakota State University. Like all things technical, a serious discussion of the issues here would not be complete without a brief look into the future. From my perspective, I can see a time where the desktop operating systems evolve to such a point where user's can manipulate their own desktop environments. To some extent this is available today but would be terribly unmanageable. When a consistent means of managing all student, faculty and staff computer accounts comes into view there will begin to be an integration of the traditional e-mail services with cluster computer resources. The idea that any user with a valid identification number could be offered a consistent set of customizable services, regardless of physical location, is something that is exciting to think about. This, of course, means continual improvement of the existing methodologies is necessary in order to learn more about what plans must be made for the future.

A Pilot Study: Offering a Web Course at a Small, Liberal Arts College

Jamie Partridge
Assistant Professor of Management
College of St. Benedict/St. John's University
37 S. College Avenue
St. Joseph, MN 56374
Jpartridge@csbsju.edu

Introduction:

This paper will detail my experiences of creating a web-based course as part of a pilot study at the College of St. Benedict and St. John's University (CSB/SJU). The impetus behind the development of this web-based course was an initiative from the academic deans during the summer of 1997 to encourage proposals to place an existing course on the web. Four professors (including myself) were awarded a faculty development grant to develop and implement a web course to serve both the Benedictine University College (in the Bahamas) and CSB/SJU (in central Minnesota). These courses were to be offered during the spring semester of 1998. The course that I received the web grant for is a qualitative and quantitative methods of management course. This course is a required course for all management majors and is designed for students to take during their sophomore year. (It is a 200 level course). This course emphasizes computer applications of business, where students learn the software tools necessary to make informed business decisions (utilizing Microsoft Office software and the Internet).

The first section of this paper will provide an overview of how to put a course on-line. The next section of the paper will include a discussion of the preliminary on-line course that I utilized in the fall of 1997 verses the more in-depth on-line course that I have developed for the spring semester of 1998. The following section of the paper will detail the pros and cons of developing and utilizing an on-line course. The next section of the paper will compare my web-based course to other web-based courses on the CSB/SJU campuses. Finally, the paper concludes with a summary of my thoughts on utilizing a web-based course at a small, liberal arts college.

How to Create a Web Course:

What is a Web Course?:

Before discussing how to make a course a web course, there needs to be some agreement as to what a web course is. According to the University of Minnesota's Teaching Assistant (TA) Web Project, course web sites can enhance teaching and learning by allowing students to access course materials, supplemental research and study materials, classmates, and instructors remotely at anytime [University of Minnesota TA Web Project, 1996]. A web course can vary from course content stored on the web in a format much like a book to content that can be served up by means of an interactive, multimedia-based instructional program or simulation [Boettcher and Cartwright, 1997]. At CSB/SJU, web courses run the gamut of instructors putting their syllabi and some ancillary materials on the web to faculty having all of their course material available on the web with web-based discussion forums, remote quizzes, etc. Thus, there is no universal agreement as to what a web course really is. Some faculty and administrators view it as a stand-alone, distance learning course where anyone can enroll. Others view it as a course where the web is used to supplement current course content.

How to Put a Course On-Line:

The most important factor in putting all or part of a course on-line is getting the proper financial support from your institution. Many institutions are offering grants to faculty members to put courses on-line. Others, like the University of Minnesota offer awards to teaching assistants for assisting professors in creating web-based courses [University of Minnesota TA Web Project, 1996]. Also, institutions must provide computers and computer support to faculty, staff, and students on campus to provide the infrastructure and training for web-based courses. For example, a political science professor at Marist College was able to provide an international relations computer simulation to his students with the help of computer support at his institution [Vavrina, 1995]. My institutions have undertaken a five-year computer improvement initiative (Project Impact) to upgrade computers on campus and provide adequate computer support. The colleges also offer training sessions and workshops on creating and supporting web pages.

Another important environmental factor that needs to be present to support web-based courses is physical spaces. [Boetcher and Cartwright, 1997]. Traditional classrooms may need to be converted into computer labs. Classrooms throughout campus may need to be wired for Internet access. Dorms may need to be outfitted with more computers.

Once all of the institutional factors are in place, one can then go about putting his/her course on-line. It makes sense to take advantage of what has already been done on your campus or on other campuses. Usually there is a group of faculty on the leading edge, who will have much of their course material on-line. Once you have checked out what others have done, you can come up with an idea of what you want to do with your course.

You should receive some training in how to put your course on the web. Student workers can be very helpful in this endeavor. Usually, a faculty member will have to obtain a Unix account before they can put material on the web. At CSB/SJU, there are several web editors available. I chose to use Claris HomePage 2.0 as my web editor. I also am able to choose map network drive in windows NT so that I can upload my .html, .GIF, .JPEG, etc. files right to my Unix account without having to use a file transfer protocol. As you can see, instructors can put course material on a web page without needing to know HTML. As indicated in a recent article in *The Chronicles of Higher Education*, both commercial and academic software developers are marketing products that enable college teachers to design World Wide Web-based courses without needing to know HTML [McCollum, 1997]. Also, with the advent of Microsoft Word 97, instructors can create assignments in Word and then save them as .html files.

Why Put a Course On-Line:

There are many reasons why a faculty member would want to put a course on-line. My biggest reason was to make material more accessible to students so that they can obtain course material at any time of the day at their convenience. By the end of the fall semester of 1997, I had my course syllabus and assignments on the web in addition to course resources (important URLs). Many other instructors put their courses on-line for this very reason. For example, a geology instructor indicates that his home page provides access to current research and reports [Childress, 1997]. A law school professor indicated that he uses the Internet to reach students in remote locations [Keeva, 1997]. At the University of Utah Medical School, web based teaching is used so that students have access to material 24 hours per day [Klatt, 1997]. These are also advantages for using the web at my institutions. Even though CSB/SJU are residential campuses with a traditional student body, students are geographically spread out since CSB and SJU's campuses are over five miles apart. It is often more convenient for students to access material from one campus or the other, rather than travelling back and forth between campuses (especially in the cold Minnesota winters). Also, CSB/SJU has an academic relationship with Benedictine University College in the Bahamas. Thus, students in the Bahamas can access course material from courses in Minnesota. This should foster more collaboration between the two campuses.

Another reason for developing a web course is to encourage more faculty-to-student and student-to-student dialogue [Boettcher and Cartwright, 1997]. A web course allows the instructor to communicate with students outside of lecture, discussion, and office hours. It also enables students and the instructor to communicate with one another at their convenience. For example, messages can be posted by students at midnight and read the following morning by the instructor [McGreal, 1997]. Students can also interact with each other in study groups and collaborative student projects. My current web course (spring 1998) incorporates more faculty-to-student and student-to-student dialogue then my previous web course (fall 1997). My new course web site now has a discussion forum where students and the instructor can communicate. Student work groups also have their own discussion forum. The students in Minnesota are also able to interact with students in the Bahamas.

A web course also allows for more student-to-resource dialogue [Boettcher and Cartwright, 1997]. This enables students to have access to course Web sites, online databases, library resources and simulations. For example, students in the Department of Meteorology at Texas A & M University use specialized meteorological software as part of a set of web-based teaching materials [Nielsen-Gammon et. al., 1996]. Also, students at Marist College use International Communication and Negotiation Simulations developed by the University of Maryland in their political science course [Vavrina, 1995]. At this point, my web course does not use simulations. However, students have access to Web sites, online databases, and library resources, which they utilize for class assignments.

Out with the Old (Web Course) and in with the New (Web Course):

Qualitative and Quantitative Methods of Management Course - Fall 1997:

I received a grant to develop a web course for the spring semester of 1998. However, I wanted to test out developing the course during the fall term. I felt comfortable creating web pages since I taught students how to create their own web pages in my qualitative and quantitative methods course during the 1996-1997 academic year. We utilized a web editor available on campus and supported by our colleges' computer network (Claris HomePage 1.0 and then version 2.0). Since the course meets in the computer lab, students were very comfortable working with the network software (i.e. Microsoft Office), e-mail (Microsoft Exchange) and other programs. Since I was already comfortable creating my own web pages and the students in the course met in a computer lab and used computer applications every day, it made sense to put course material on the Web.

During the fall semester, I created a simple course web site. I listed the course name on my home page. Then, I had listings of course material available on the web with links. I put my course syllabus as a link. I put all class assignments as links. I also included a link to a listing of URLs that students could use for assignments - i.e. business and economic information. Finally, I had links to pages, which listed student work groups and other material. Assignments were easy to put up on the web because all I had to do was save my Word files as .html files and then move them to my Unix account. Since I could map to my network drive in Windows NT instead of uploading all of my files (using FTP) to my account, this process was very quick and easy.

I worked with another faculty member in my department who was also teaching this course and I helped her develop a similar web site. This enabled us to offer some consistency across sections of the course. However, a drawback was that I often found myself doing double the work. It would have been easier for us to have a common web site (or for us to work completely independently).

By the end of the semester, I was happy to have my course web site up and running. However, I was disappointed that it did not appear to enhance the effectiveness of the course. Since I provided students with handouts on all assignments, there was no incentive for students to use my web site to look up assignments. Also, I put assignments up at the same time that a hard copy was passed out. A better approach would be to provide all assignments on the web at the beginning of the semester (and not to provide hard copies). By the end of the semester, students did utilize my course resource site (with links to other sites). Since I was building the site as the semester progressed, it was not complete until the end of the semester. In the spring, this site will be up and running from day one.

My biggest disappointment with my web site was that students weren't able to participate in discussion forums. The only way we communicated electronically as a class was via e-mail. I set up a class distribution list and e-mailed students assignments (i.e. messages with file attachments). They could then e-mail homework to me (with file attachments). However, we did not engage in any on-line discussion or dialogue as a class. Microsoft Exchange (our e-mail system) does have public folders where classes can engage in discussion. I set up a public folder for the course, but no one wanted to utilize it since it was awkward. In the spring semester, I have implemented discussion forums into my web site.

Finally, I believe that a major purpose of a web site is to provide students off campus access to course material at any time of the day. I believe that my web site did this. However, I felt that students from the Bahamas should be able to also utilize this material. Students from the Bahamas usually spend one semester at CSB/SJU to complete their degree. Since Benedictine University College (BUC) in the Bahamas does not have the equivalent computer applications course for management students, many management students come to Minnesota without the appropriate computer skills for upper-level management courses. It would be beneficial to the students in the Bahamas to have access to this course material before they come to Minnesota to take upper level management courses. During the spring semester, one student in the Bahamas at Benedictine University College is taking the course via the web.

Qualitative and Quantitative Methods of Management Course - Spring 1998:

I have completely revamped my web course for the spring of 1998. I decided to utilize *Web Course in a Box* (developed at Virginia commonwealth University) as a web-authoring tool. According to the user documentation for *Web Course in a Box* (WCB), WCB is a course creation and management tool for Web-based or Web-assisted delivery of instruction. It enables the creation of basic course pages such as a syllabus, class schedule, personal home pages, as well as interactive Web functions such as discussion forums and self-correcting exercises. Authoring is done using a Web browser and does not require any knowledge of HTML or other technical skills [Godwin-Jones and Polyson, 1997]. WBC offers on-demand access to learning materials, integration of local and global resources, up-to-date information, collaborative learning, and multi-media presentation of content [Godwin-Jones and Polyson, 1997]. The last two features were what appealed to me, since my previous web site did not provide these capabilities.

Students access my *Web Course in a Box* Web site from my own home page: http://www.users.csbsju.edu/~jpartrid. My home page provides basic information about my institution and myself and includes the courses that I teach. Students can select my qualitative and quantitative methods course to link to the *Web Course in a Box* Web site. There are certain features of the site that are password protected so that only students (or guests) can access the site. For example, to participate in the discussion forum or to take an on-line quiz, you need a username and password. The Web site is set up so that students can select various pictured icons to get course information or to interact with one another. The following categories are part of the *Web Course in a Box* course pages for students:

Class Info:

Students can select course information to link to general course information, a course description, the required text, and to my syllabus (which I created in Word and saved as an .html file). Students can click on pictured icons here to go back to the *Web Course in a Box* home page or to my home page. Students can also e-mail me from here. These features apply to all of the other categories as well.

Announcements:

A student can check any important announcements from the instructor. This alerts students of any up-to-date information. If there is an announcement, this icon flashes on the computer screen. This feature could replace the instructor e-mailing the class (via a distribution list) when he/she needs to provide students with some information.

Schedule:

This feature enables students to look at the course schedule in an organized fashion. I listed daily lessons for each week of the semester and students can select any week to view a week at a glance.

Students:

This section includes a student directory where students or the instructor can select to e-mail the entire class or an individual in the class.

Learning Links:

This is the most in-depth section. This section includes discussion forums, lessons, exercises, and quizzes. The instructor sets up a discussion forum and students, guests and the instructor can post messages to the topic. Messages are dated and time stamped. Examples of discussion forums include students providing their computer background to the class and students discussing problems that they are having completing an assignment. Students need a username and password to participate in discussion forums and each student is identified by his/her username. Users can also include a file attachment with their messages in a discussion forum. However, I was only able to leave a Word attachment (.doc) and not an Excel attachment (.xls). Computer support staff are working to fix this problem. If the Excel attachment feature is fixed, then I can provide students with practice spreadsheet files from my web site instead of e-mailing them file attachments (via a distribution list). The instructor also has the ability to delete messages from the discussion forum by username, date, etc.

Under lessons, one can list projects and exercises for students to access. I have this section organized by projects. Students can select a project and then link to several options. For example, they can link to practice exercises or they can link to an assignment to turn in. The course is organized into 10 such projects (modules). I used my previously created Word files (and saved them as.html files) and linked these to my projects. I also included links to other course information including course resources (which list URLs that students cans use to obtain economic and business information for assignments).

The instructor can also develop exercises and quizzes (which are password protected) for students to take over the computer. I use this section for students to take practice quizzes on the material. The quizzes can be set up from an instructor's web site. The quizzes are in a multiple-choice format. Students can retake quizzes several times and the results are submitted to the web page. This enables students to get quick feedback on how well they know the material.

Help/Utilities:

This section enables students to change their passwords and to create/edit their own homepage. I will not use this homepage feature for students in the course. I will have students create their own home page using Claris HomePage 2.0. All students will receive their own Unix accounts. They can then link their home page to the course page at the end of the semester.

Other Features of Web Course in a Box:

In addition to the WCB course pages for students, *Web Course in the Box* has a WCB server (for systems administrators) and WCB authoring tools (for faculty). Presently, the WCB server is located in our colleges' computer science department. With more and more faculty at my institution utilizing WCB, the server will have to be moved eventually to the computer center. Creation of web pages using WCB is done entirely over the Web using a Web browser (i.e. Netscape Communicator).

The WCB authoring tools for faculty consist of an integrated series of Web forms which enable instructors to create and update Web pages for classes, manage course and student information, create discussion forums, and upload and organize course content files. Access to WCB authoring tools is limited to instructors. Editing pages for the instructor includes a WCB menu, information on students, utilities, feedback and other info.

The WCB menu is divided up into sections. The instructor can select preferences, where he/she can restrict access to the web page in addition to changing the look of the web page. The instructor can also enter information here under class info, schedule, announcements, and learning links (which include forum builders, lesson builders, and quiz builders). This information then appears on the student web pages. The instructor enters information on students in another section. The instructor can also edit this information. The utilities section lets instructors edit certain information. For example, this is where the instructor can edit discussion forums. The feedback section links the instructor to other web sites to provide feedback to other users. Finally, the other information section allows the instructor to receive more information on *Web Course in a Box* (i.e. an instructor's guide, a student guide).

In addition to utilizing WCB during the spring semester, I have also taken on a student who is taking my web course as a stand-alone course from the Bahamas. This student is affiliated with Benedictine University College. He accesses all of the course material via the web and turns in assignments via Microsoft Exchange using file attachments. This setup seems to be working well for the student since he is a nontraditional student who has a full-time job with the telephone company in the Bahamas. He is able to do much of his work from home using his own computer. This fulfills a common aim of distance education programs to reach learners who would otherwise not be students [Simonson, 1997].

The Old (Web Course) Versus the New (Web Course):

There are many advantages of using the *Web Course in a Box* as a Web authoring tool. Students can utilize discussion forums to interact with one another as well as the instructor. Also, students in the Bahamas at Benedictine University College (BUC) can participate in class discussion with students from CSB/SJU in Minnesota. Another advantage of *Web Course in a Box* is that access can be restricted to just those enrolled (or guests) in the class. This provides a safe forum for students to participate in discussion forums. A further advantage of *Web Course in a Box* is its quiz builder capabilities. Finally, Web Course in a Box provides the student with a convenient interface to access course materials in an organized fashion.

Web Course in a Box does appear to have some drawbacks. For example, the interface cannot be customized very well unless the user is familiar with HTML or a web editor. Thus, for a novice user, WCB cannot be easily changed. There also appears to be some glitches in the software for WCB. For example, I am not able to attach an Excel file to a message that I post in WCB for students to access. Finally, although restricted access has benefits, it also has drawbacks in that in order to participate in discussion forums and take quizzes, you need a username and a password. In conclusion, I will be able to better evaluate the *Web Course in a Box* authoring tool once the semester is over.

Evaluation of the New Web Course:

I plan on having a formal assessment of the course done twice during the semester (halfway through and at the end of the term). This assessment will take the form of an online questionnaire where students can enter responses anonymously. Additionally, I plan on having one of our in-house software experts from information technology assess the usability of my web page.

The Pros and Cons of a Web Course:

Advantages of a Web Course:

There are numerous advantages to creating a web-based course. One advantage mentioned previously is the web's ability to provide students important course information 24-hours-a-day any place in the world that has access to the Internet. This is particularly important for the non-traditional student population that lives off campus. This is also important for students who live in rural areas. At CSB/SJU, the advantage here is that students can access information from either campus at any time. Also, students in the Bahamas can access this information (or even take the course as one student is doing now). Finally, students that are involved in study abroad programs can also have access to course material on the Internet. For example, a psychology professor at CSB/SJU is doing a study abroad program in South Africa during the Spring Semester of 1998 and he is using course material that is available on his web site.

Surprisingly, a Web-based course can also promote more teacher-student interaction [Childress 1997, Boettcher and Cartwright, 1997] and student-student interaction [Boettcher and Cartwright, 1997]. Students often feel more comfortable communicating with the professor and other students electronically. Often, the less outgoing students who would be reluctant to participate in face-to-face class discussion are more comfortable participating in an electronic forum. Also, a Web-based course serves the function of extending the classroom walls beyond scheduled class meetings and office hours. It adds a lot more flexibility to the learning environment.

As mentioned before, another advantage of a Web-based course is that it provides students access to current research and reports [Childress, 1997]. The instructor can come up with a list of URLs that he/she has previewed to ensure that they are up to academic standards. The instructor can also have links to academic databases, library resources, and government data sites. Also, the course's Web page can have links to students' work, so that students in the class can view each other's work.

Instructors can also achieve "economies of scale" by creating a Web-based course. For example, instructors can share resources for an introductory course with multiple sections. The economics department at CSB/SJU is doing this with their principles of economics course during the spring semester of 1998. Faculty members are compiling a common body of information for students to utilize (including course material, links to economic data, etc.). Some instructors have even found that their time in the classroom can be reduced with the availability of course material on the Web. Pathology faculty at the University of Utah's Medical School found that they had a 30% reduction in classroom instructional hours since they made more course material available on the Web [Klatt, 1997].

Finally, another advantage of providing a Web-based course is the ability for students to participate in simulations or other forms of interactive learning. Students at Marist College utilize a Web-based Model United Nations simulation in an international relations class [Vavrina, 1995]. Students at Texas A & M utilize computer-based software in a meteorology course. Medical students at the University of Utah utilize Web-based course material to view pictures of lab specimens that have been uploaded onto the Internet. The applications are limitless for this type of technology.

Disadvantages of a Web Course:

There are also disadvantages to developing a Web-based course. First, it can be very time-consuming to get the course up and running. A lot of time needs to be invested on the front end. It is particularly difficult for faculty that have very little computer experience. Also, many institutions do not reward faculty for developing Web-based courses. It is interesting to note that the University of Minnesota rewards teaching assistants for excellent Web Courses, but does not reward faculty members [University of Minnesota TA Web Project, 1996].

Another disadvantage of a Web-based course is that there may be less classroom interaction. This is particularly true with distance learning or courses that do not meet on a regular basis. In particular, many courses are best left to current classroom discussion format. A further danger is that distance learning may promote the use of better-known professors at the expense of lesser-known professors. This is a concern of a law professor who believes that teaching law over the Internet may jeopardize direct classroom interaction as well as promote better-known professors over lesser-known professors [Keeva, 1997]. A related concern is property rights over an instructor's course material that is posted to the Internet. Does the college or the professor "own" the material?

Finally, a Web-based course might inadvertently sanction the use of only Internet sites as sources of information. This can be dangerous. Many sites on the Internet are excellent (i.e. journal articles, government sites, and databases). However, many students have the tendency to type in a key word into a search engine like Yahoo! and use the information (or misinformation) that they come up with in a report.

Other Web Courses at CSB/SJU:

Three other departments at CSB/SJU received grants to develop Web Courses for the spring of 1998. Five faculty members from the economics department received a grant to develop a Web-based course for the department's principles of economics course. They have decided to use *Web Course in a Box* as their web-authoring tool. They are using the Web to supplement (not replace) their class meetings. For example, text material, problems, self-quizzes, Web links and discussion forums will be part of their Web site.

Aubrey Immelman of the psychology department received a grant to develop a Web-based course in group dynamics. Professor Immelman is utilizing a *Web Course in a Box* as his web-authoring tool. Professor Immelman will be teaching this course during a study abroad in South Africa in the spring of 1998. Professor Immelman's textbook for the course is only available on the Web (i.e. not in hard copy format)..

Kaarin Johnston of the theater department received a grant to develop a Web-based course in modern drama. Professor Johnston is utilizing a *Web Course in a Box* as her web-authoring tool. Professor Johnston has 12 students enrolled in her course. Her Web course is a stand-alone course, where the only interaction is via the Web (i.e. there are no class meetings scheduled). Professor Johnston believes that her Web-course will foster better writing skills among her students.

Other faculty members at CSB/SJU are incorporating the Web into their courses. For example, Kathy Twohy of the nursing department is using the Web to supplement her leadership in nursing class in the spring of 1998. There will be three students (out of 50) that will not attend class sessions and will take this nursing class utilizing the Web site only. Tom Creed of the psychology department utilizes a *Web Course in a Box* software to teach a principles of learning and behavior class as well as a popular delusions class. Students participate in discussion forums via his web site. Much of professor Creed's material is available via his Web site.

Conclusion:

In conclusion, I would like to share some thoughts on utilizing a web-based course at a small, liberal arts college. I believe that the Web is an excellent source of supplementary material for students. However, the Web should primarily be used in conjunction with a traditional class lecture/discussion format rather than as a stand-alone course with no class meetings. The Web just introduces another way for students to connect to the professor, other students, and more information. It is not a substitute for direct contact with the instructor and other students. However, a stand-alone web course may work well for non-traditional students at remote sites (i.e. students in the Bahamas at BUC). Also, students shouldn't be led to believe that the only information out there is available on the Web. Libraries have not become obsolete yet!

A Web-enhanced course can be a better course than one without the Web. However, it takes a lot of time and effort to come up with a good Web-based course. Institutions must reward faculty members for taking the time to "upgrade" their courses.

On college campuses such as CSB/SJU, almost all of the students are traditional students and thus stay on campus. This enables the instructor to be confident that students have sufficient computer access 24-hours per day. At larger state institutions, some students may not have enough access to computers to obtain course information. Thus, small, liberal arts colleges have a strategic advantage in this area.

Finally, today's college students are more comfortable using the computer as a learning tool then are most faculty members. Even if we don't like it, this is how most of our students are comfortable interacting and receiving information. We should take advantage of the Web by incorporating it into our classes in a useful manner, rather than letting the opportunity pass us by.

References:

Boettcher, J. & Cartwright, G. (1997). Designing and Supporting Courses on the Web. Change, 29 (5), 10-13.

Childress, J. (1997). Web Offers New Learning Options. Geotimes, 42 (1), 11.

Godwin-Jones, B. & S. Polyson. (1997). Web Course in a Box User's Guide Version 2.0. Virginia Commonwealth University. http://www.madduck.com/wcbinfo/wcb.html.

Keeva, S. (1997). Stars of the Classroom: Will top Profs who Instruct via Internet Dominate Teaching? ABA Journal, 83 (Dec.), 18-20.

Klatt, E. (1997). Web-based Teaching in Pathology. The Journal of the American Medical Association, 278 (21), 1787.

McCollum, K. (1997). A New Industry Sprouts up to Help Professors put courses on Line. The Chronicle of Higher Education, 44 (10), A33-34.

McGreal, R. (1997). The Internet: A Learning Environment. New Directions for Teaching and Learning, (71), 67-74.

Nielsen-Gammon, J., Biggerstaff, M., Alcorn, M., Austin, D., Bowman, K., Djuric, D., Guynes, J., Panneta, R., White, R., & Wicker, L. (1996). Texas A&M University's Laboratory for the Exploration of Atmospheric Processes - TAMU's LEAP. Bulletin of the American Meteorological Society, 77 (12), 2907-2918.

Simonson, M. (1997). Evaluating Teaching and Learning at a Distance. New Directions for Teaching and Learning, (71), 87-94.

The Web Project 1996-1997 (1996), University of Minnesota Digital Media Center, Minneapolis, MN.

Vavrina, V. (1995). Poughkeepsie to Persian Gulf Revisited: ICONS, the Internet and Teaching International Politics. Political Science and Politics, 28 (4), 725-729.

Acknowledgements:

The author would like to thank St. John's University and the College of St. Benedict for the $2000 stipend to develop an on-line course (qualitative and quantitative methods of management).

ALGORITHM ANIMATION USING JAVA

Dean M. Patten
Undergraduate Research Project
Department of Computer Science
University of Northern Iowa
patten@cns.uni.edu

This paper reports my undergraduate student research project. My topic is algorithm animation of optimization problems solved using backtracking and branch-and-bound. Java was chosen to display the algorithms graphically because of the simplicity of its graphical classes and its easy integration in web pages. My algorithm animation project is designed to run over the Internet, and is supported by all major web-browsers that are out today. The project is designed to interact with the user and to stimulate his/her knowledge of backtracking and branch-and-bound. The final product will be used in certain university classrooms to help instructors teach students these problem solving techniques. If teachers have the ability to show these types of algorithms via animation in their classroom, students will gain valuable insight to backtracking and branch-and-bound.

ALGORITHM ANIMATION USING JAVA

Dean Patten
(Undergraduate Research Project)
Department of Computer Science
University of Northern Iowa
patten@cns.uni.edu

I. Introduction

The problem that I am handling originated in my “Design and Analysis of Algorithms” class in the spring of 1997. My professor and textbook [Neapolitan 1996] were attempting to describe the backtracking algorithm to handle an optimization problem. I was having trouble seeing the algorithm at work, as the professor was drawing the data set as a tree. Many of the students in the class could understand the artistry just fine, which confused me more. After pondering the problem for the evening, I found that by looking at a matrix representation of the data set I could easily see the transactions that the optimization algorithm was performing. At the next class, I brought up the idea and tried to explain it by drawing it on the board. Now this was interesting. The individuals that understood the tree representation of the data set did not see this new proposed view, while the students who had trouble with the first view could easily follow my representation. It would seem that people view this type of problem in as least two different ways.

I decided to act on this information, and dedicate my research project to this type of study. I was more concerned with the implementation of optimization algorithms that could be viewed in a couple different ways than why students picked up on one view or the other. I wanted a way for students, like myself, who did not follow the conceptual ideas of the instructor, to understand optimization algorithms from a different point of view. I feel that this would help the student learn the structure of the algorithm and follow it through to completeness.

In addition to this ultimate goal, I also had some personal motivations. I was very interested in learning the language Java. I believe that Java will become an important entity in the future of computers. I also wanted to completely understand a few optimization algorithms like backtrack and branch and bound. It is easy to glance over the psuedocode for an algorithm and not completely understand it. By programming these algorithms, I hoped to get in-depth view of recursion and its effects on the data set.

Trying to depict the operations of an optimization algorithm on a multidimensional array can and is often confusing. An array of size, say, 3 by 3 would not be too hard to grasp as it only has 3 different and distinct paths, but what about a 5 by 5 array? Here, this array would have about 150 different paths. The problem is exponential. It quickly becomes very challenging for students to depict these different paths. My research is based on this challenge.

At the onset of the project I had a choice to make, how to graphically depict the optimization algorithm. I initially debated between Visual C++ and Java. Based on the statement I listed above, I decided to pursue implementation in Java. Now I had a new decision, how to view the project, as a standalone application or as a web page applet. Since I was looking for an easy way for students to access my research I decided to push my research web side. I setup my personal computer as a web-server and published my pages that way. This has also allowed me to monitor who is viewing the pages and for how long. From this I can see how useful the project has been.

II. Animated Algorithm Analyzer

The project timeline only allowed me time to animate a single optimization problem, the Job-Assignment problem, using two different problem solving techniques: Backtracking and Branch-and-Bound. The Job-Assignment optimization problem was chosen to be animated because it cleanly illustrated the matrix representation that helped me understand backtracking. The key characteristic of the Job-Assignment problem that allows the matrix representation view of backtracking is that a solution consists of selecting only one element from each column and row of the data set in matrix representation.

Handling this type of problem requires some pondering. In designing the project, I came up with an example similar to the following to try and get a better understanding of the Job-Assignment problem. Let us say that there are three people, Tom, Sally, and Phil, who all have a certain skill that they excel in. You being the manager of the three must set up a work schedule that will optimize the time spent for each job. You are handed a schedule similar to Figure II.1 that lists their name, skill, and the amount of time in minutes required for them to finish that task. Keeping in mind that all the individuals are capable of doing any job, it is now your responsibility to set the schedule.

Name	Network Wiring	PC Configuring	Break/Fix
Tom	10	9	15
Sally	15	11	8
Phil	12	13	9

Figure II.1: Time to complete tasks.

The idea is to minimize the time spent in each given position. Looking at the Network Wiring position, it would appear that Tom is the ideal choice at an average of 10 minutes. Then by applying this same method to the next positions, we find that Sally would be the next best pick for the PC Configuring position, which leaves Phil for the Break/Fix position. This path through the data set has a solution value of 30 minutes (10+11+9). Is this the Job-Assignment's optimal solution? The answer is no. Even though Tom is the best choice for Network Wiring, he is an even better choice for PC Configuring. Now, since the PC Configuring position is filled, only Network Wiring and Break/Fix are left. Sally best fits in Break/Fix with an average of 8 minutes, while Phil is in Network Wiring with an average of 12 minutes. This path through the data set has a solution of 29 minutes (9+8+12). This answer is the optimal solution for the 3x3 Job-Assignment problem. Finding the solution was easy. Could you imagine searching for a solution to a 5x5 Job-Assignment problem? Figure II.2 shows the optimal solution in table format for the 3x3 Job-Assignment problem.

Tom	PC Configuration
Sally	Break/Fix
Phil	Network Wiring

Figure II.2: Optimal job assignment.

The understanding of this principle is crucial in understanding the algorithms used to obtain the optimal solution.

The project timeline allowed me to implement two algorithms, namely Backtracking and Branch and Bound. The flow chart of the project is shown in Figure II.3. The main page is the initial start state. There are links to the research paper and to each algorithm homepage. From here the project forks depending on the algorithm selected for viewing. Each algorithm path utilizes an algorithm analyzer, which is where the implementation of the algorithm takes place. I will describe the Algorithm Analyzer in more detail in the following section. By using web pages as my underlying design, the flow of the project is straightforward. Every consecutive section is accessible via a web link.

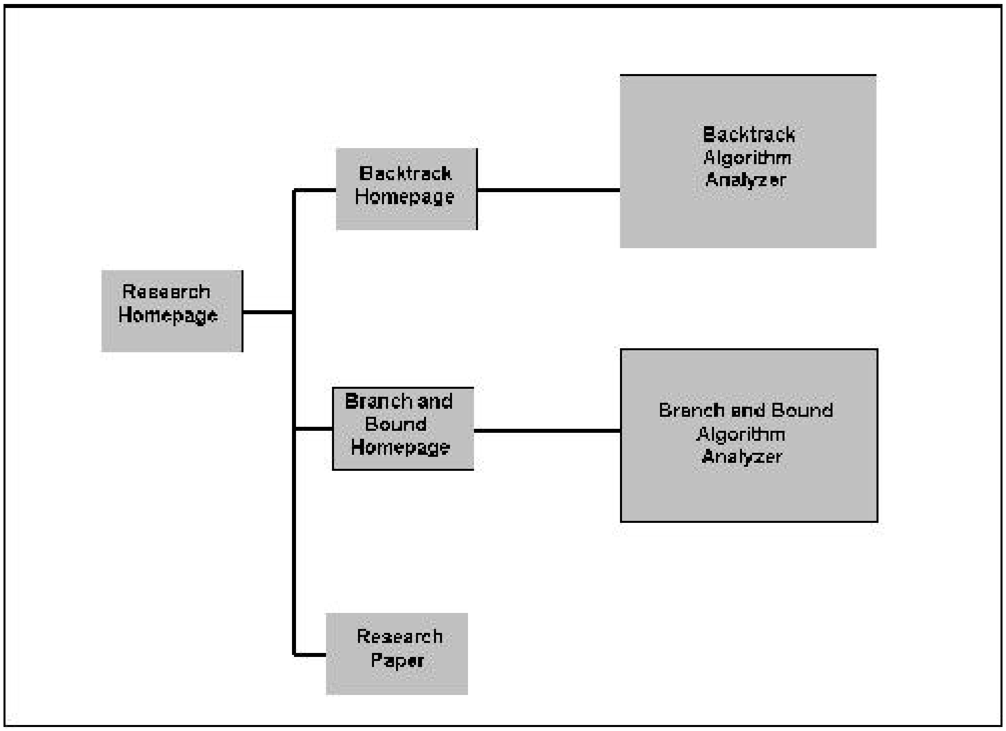
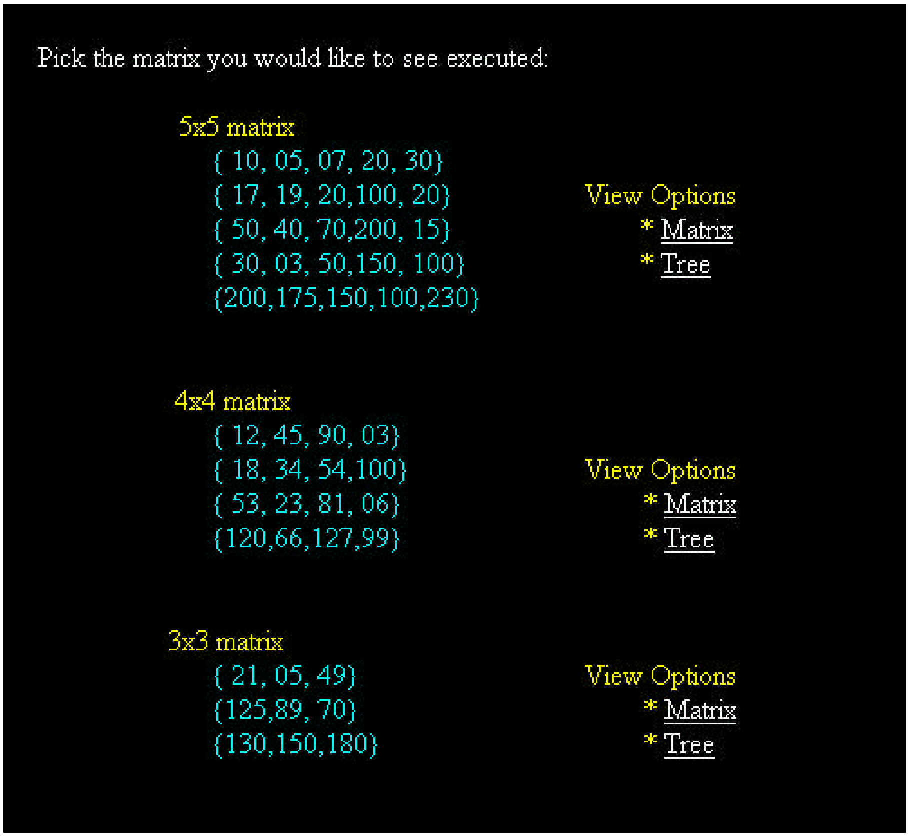

Research Homepage
Backtrack Homepage
Backtrack Algorithm Analyzer
Branch and Bound Homepage
Branch and Bound Algorithm Analyzer
Research Paper

Figure II.3: Hierarchy of my research web pages.

To graphically represent the algorithm at work, I first needed to decide how to depict the different views. I initially designed the Algorithm Analyzer to handle only the matrix approach on depicting the algorithm. It was only after the completion of that task did I implement the traditional search-space tree view. I also wanted to allow the user to enter the data set used for the Algorithm Analyzer. After a terrible battle with perl scripts, I abandoned the approach and chose a simpler design. This design required the data sets to be hard coded into the web pages. Although there is an option of several different data sets to chose from, it is not as robust as I would like this portion of the project to be. Figure II.4 shows the selection options from the backtracking section.



Pick the matrix you would like to see executed:

5x5 matrix

```
{ 10, 05, 07, 20, 30}
{ 17, 19, 20,100, 20}        View Options
{ 50, 40, 70,200, 15}            * Matrix
{ 30, 03, 50,150, 100}           * Tree
{200,175,150,100,230}
```

4x4 matrix

```
{ 12, 45, 90, 03}
{ 18, 34, 54,100}            View Options
{ 53, 23, 81, 06}                * Matrix
{120,66,127,99}                  * Tree
```

3x3 matrix

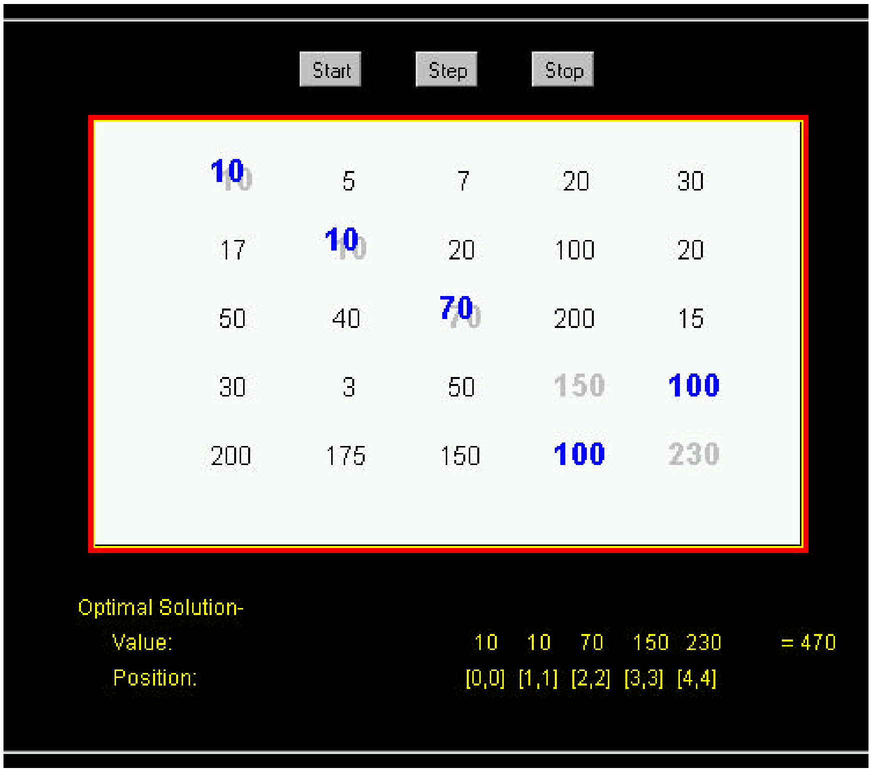
```
{ 21, 05, 49}                View Options
{125,89, 70}                     * Matrix
{130,150,180}                    * Tree
```

Figure II.4: Available data set options.

I wanted to represent several different data sets and finally chose the data sets of a 5x5, 4x4, and a 3x3 matrix. I felt that this would give a good idea of the algorithms performance. It is quite interesting to see the amount of time to complete the algorithm for the different matrixes. The actual data elements in these sets were chosen at random.

From this point, the user is able to pick the view and the data set that they would like to see run through the Algorithm Analyzer. Figure II.5 shows the matrix approach to the 5x5 data set. Figure II.6 depicts the same 5x5 data set, but in the traditional search-space tree view.

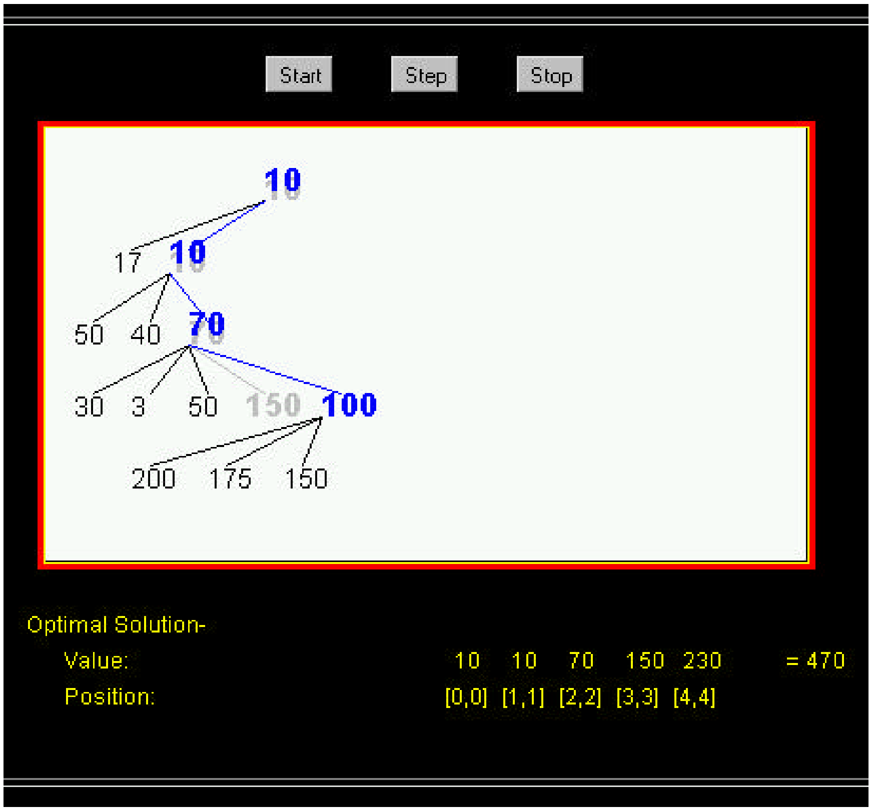
Both figures represent approximately the same point in the evolution of the algorithm, but by the way they are presented, show completely different things. These different views are the most common ways that students approach the optimization problem. I have designed the Algorithm Analyzer for the ease of understanding the basic underlying principle, the optimal solution. I used colors to represent the states the algorithm was currently at. For example, blue indicates an active state. The active state is the potential solution that the Algorithm Analyzer has built at the current time. Once a solution is found, the positions and the values of the solution are displayed in the summary statement, which is listed below the active window. The solution set's color is then changed to gray, to represent that an optimal solution has been found. To indicate that an element is both an active state and a solution, I chose to superimpose blue over gray as shown in both the figures.



10	5	7	20	30
17	10	20	100	20
50	40	70	200	15
30	3	50	150	100
200	175	150	100	230

```
Optimal Solution-
   Value:                       10    10    70    150   230       = 470
   Position:                   [0,0] [1,1] [2,2] [3,3] [4,4]
```

Figure II.5: Matrix view of the 5x5 data set.



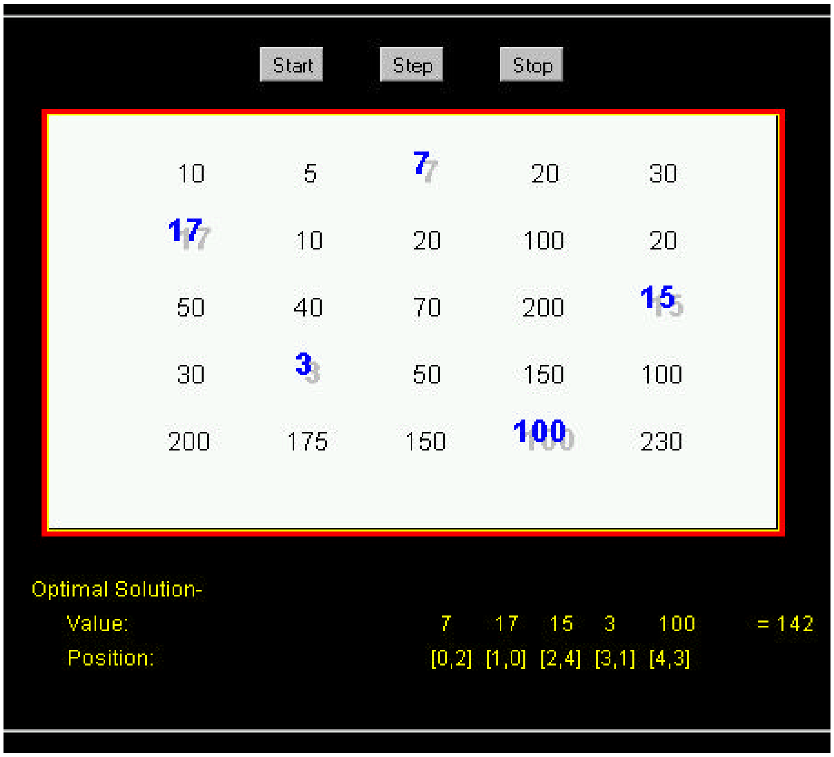
```
Optimal Solution-
   Value:                       10    10    70    150   230       = 470
   Position:                   [0,0] [1,1] [2,2] [3,3] [4,4]
```

Figure II.6: Tree view of the 5x5 data set.

It is very apparent in the matrix approach that the current solution of {10,10,70,150,230} with a value of 470 has been reached. From here, it is also determined that the active state is the set {10,10,70,100,100}. In the tree representation, the solution to date is not as clear. The solution is listed in the summary statement, but the complete solution is not visible since, in this case, the backtracking algorithm has moved up, over, and down from the solution set. It is also not as clear here what the active set is since the last row does not list its solution. Actually the tree diagram from the figure is one step before the matrix diagram. If it were to traverse to the next element, which would be 100, it would mark that as the active state and be the same representation as the matrix diagram.

To allow for user friendliness, the Algorithm Analyzer called for some options for pausing, stopping, and restarting the algorithm as well as keeping a running total of the solution. I implemented the three buttons that are present above the active window to handle these cases. A user can step through the algorithm by clicking on the step button. Clicking on the step button will process the next element in the data set. The start button will resume the Algorithm Analyzer to full speed from either a paused or stopped state. Finally the stop button will stop a running algorithm so that the current state can be viewed or to allow for a step transaction. The summary statement provides the current solution of the data set.

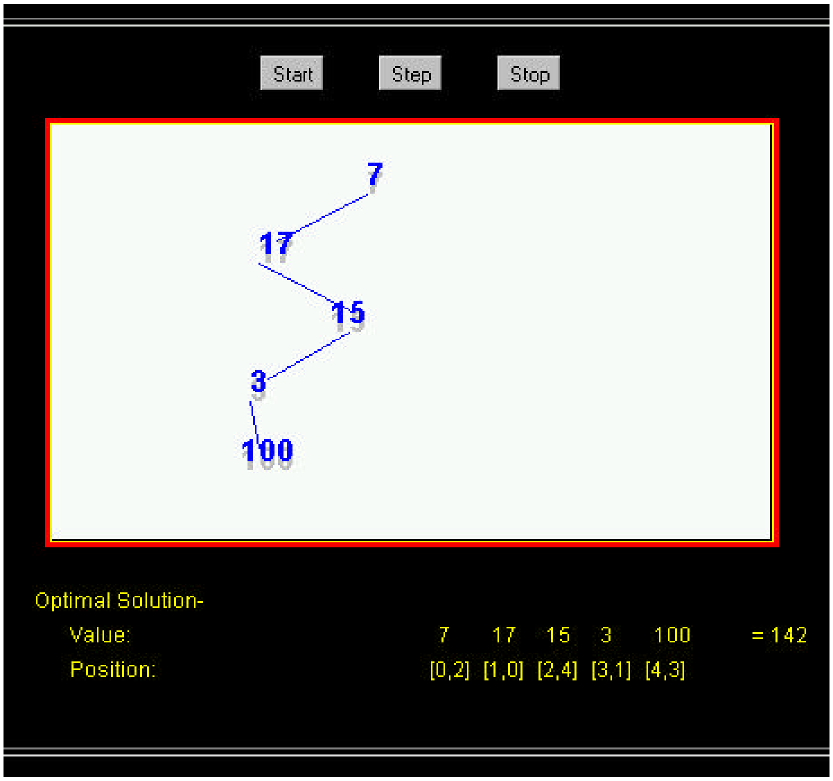
Once the Algorithm Analyzer completes, the optimal solution is displayed to the user. Depending on the view selected the user will see a different layout for the optimal solution. In the matrix view, the optimal solution is superimposed over the entire matrix. This allows the user to easily see the solution in terms of columns and rows. This graphical type of solution would be perfect for the job-assignment problem that was listed above. The tree view is not quite as simple. The optimal solution here is the only path that is listed. This was utilized for simplicity, as it would be very difficult to graphically expand all of the elements at every level of the data set. Figure II.7 shows the final optimal solution viewed as a matrix for the Backtracking Algorithm Analyzer for a 5x5 matrix given a defined data set. Figure II.8 shows the final optimal solution viewed as a tree for the Backtracking Algorithm Analyzer for a 5x5 matrix given a defined data set. We can see that both views are representing the same optimal solution with a value of 142.



10	5	7	20	30
17	10	20	100	20
50	40	70	200	15
30	3	50	150	100
200	175	150	100	230

```
Optimal Solution-
   Value:                  7     17     15     3      100        = 142
   Position:             [0,2] [1,0] [2,4] [3,1] [4,3]
```

Figure II.7: Matrix view of the optimal solution for the 5x5 data set.



```
Optimal Solution-
   Value:                     7     17    15    3     100        = 142
   Position:                [0,2]  [1,0]  [2,4]  [3,1]  [4,3]
```

Figure II.8: Tree view of the optimal solution for the 5x5 data set.

III. Implementation

The implementation of the Algorithm Analyzer was completely done in Java. It was designed to be as simple as possible but complete. I had some prior knowledge of Java from a different class, which allowed me to work with some of the more advanced features of this language.

The passing of parameters was one of the first topics that I handled. I needed a way to pass parameters from a web page to Java code. Using one of the built in classes of Java that handle parameter passing of strings, I passed the size, which view the user wanted to see, and which data set to use. Figure III.1 shows the code that was used to accept the parameters from the html page. I chose to represent these parameters as integers, each value with its own special meaning.

Since the optimization problem required recursive code I created a few global variables, but left most of them to be passed each time the function was recursively called. Figure III.2 shows the variables that are used to call the backtracking algorithm. The level represents which level in the data set is the algorithm. Jobs is a boolean array used to keep track of which jobs are not taken. Children is also an array that contains the data elements of the current level. Optimal is the current solution, and dataStatus is the state of the world, if you will. It is the current state of the data set. It keeps track of the active state, and any solution state.

By using the off screen canvas, it allows the computation of the data to happen in the background and only the finished product pasted to the screen. Figure III.3 shows the function called used paint to the off screen canvas. Depending on the view that the user selected, the appropriate procedure will be called.

```
//Get Parameters for applet
        String s = getParameter("n");         // Get the size of the matrix
        if (s == null)
                n = 3;
        n = Integer.parseInt(s);
        matrixSize = n;

        s = getParameter("PrintMatrix");    // Get boolean to see if matrix or not
        int tmp = Integer.parseInt(s);
        if (tmp == 1)
                printMatrix = true;
        else
                printMatrix = false;

        s = getParameter("matrix");         // Get which matrix we will be working with
        tmp = Integer.parseInt(s);
        data = new int[n][n];
        if (tmp == 1)
                SetUpData(data1);
        else if(tmp == 2)
                SetUpData(data2);
        else if(tmp == 3)
                SetUpData(data3);
```

Figure III.1: Code to accept parameters from html pages.

```
        Optimal = Breakdown(level, jobs, children, optimal, dataStatus);
```

Figure III.2: Backtracking parameters.

```
                Public void PaintOffScreen(int[][] dataStatus)
               {
                  if(printMatrix)
                       paintOffScreenMatrix(dataStatus);
                  else
                       paintOffScreenTree(dataStatus);
               }
```

Figure III.3: Function call to paint to off screen canvas.

Figure III.4 shows a portion of the code for painting to the virtual canvas for viewing as a tree. Here, the dataStatus array is used to get the current status of the world. The program draws the active state, current solution, and unselected states in their respective colors depending on the value that is stored in each position of the array.

```
If(dataStatus[row][col]==9)                    //Check if a solution
{
        offScreenGraphics.setColor(Color.lightGray);  //or just being checked
        offScreenGraphics.setFont(fontType2);
        offScreenGraphics.drawString(strData[row][col],xstart, ystart);
}
else if(dataStatus[row][col]==1) //adjust color accordingly
{
        offScreenGraphics.setColor(Color.blue);
        offScreenGraphics.setFont(fontType2);
        offScreenGraphics.drawString(strData[row][col],xstart, ystart);
}
else if(dataStatus[row][col]==2)
{
        offScreenGraphics.setFont(fontType2);
        offScreenGraphics.setColor(Color.lightGray);
        offScreenGraphics.drawString(strData[row][col],xstart, ystart);
        offScreenGraphics.setColor(Color.blue);
        offScreenGraphics.drawString(strData[row][col],xstart-5, ystart-5);
}
else
{
        offScreenGraphics.setFont(fontType1);
        offScreenGraphics.drawString(strData[row][col],xstart, ystart);
}
offScreenGraphics.setFont(fontType1);
offScreenGraphics.setColor(Color.black);      //reset the color
if(length == 1)                                //Realign for the next number
        xstart = xstart-10;
else if(length == 2)
        xstart = xstart-5;
```

Figure III.4: Code for painting to the virtual canvas for viewing as a tree.

The backtracking and branch and bound algorithms have similar characteristics as well as some interesting differences in their implementation. Both algorithms are recursive, use a jobs array, and manipulate their data using the dataStatus array. That is where the similarities end.

The idea of the backtracking algorithm is to find the minimal optimal solution by visiting all the combinations in the data set. This is time consuming and very inefficient. First, it descends down the data set, using the Jobs-Assignment criterion described above, until it reaches a terminal element, which means it can not go any further. At this point, a possible solution has been found. It is compared to the current optimal solution. If the new solution is smaller than the current solution, it becomes the new optimal solution. At this point, the algorithm backs up a level, moves over to the next element, and checks to see if it matches the Job-Assignment criterion. If it does, it expands upon that element until it reaches a terminal state. If the criterion does not match, the algorithm continues to move along checking all the elements it encounters. This continues until all the data set combinations have been checked.

The branch and bound algorithm is much more efficient in its handling of the data set, but requires more complex code. The idea of the branch and bound algorithm is to find the minimal optimal solution by calculating a value that is called the bound from the current optimal solution. The algorithm uses the bound by comparing itself to any future calculated values associated with the expansion of the data set.

```
Backtrack(level, jobs, children, Optimal, dataStatus)
{
        //save old data
        bestOptimal = Optimal;
        oldJobs = Jobs;
        oldDataStatus = dataStatus;
        for(0 < x < matrixSize)
        {
                if (promising)
                {
                        Jobs[x] = true;
                        Optimal[x] = level;
                        if (jobsFilled)
                        {
                        return Optimal;
                        }
                }
                else
                {
                        set up next children
                        newOptimal = Backtrack( level + 1, jobs, nextChildren,
                        Optimal, dataStatus);
                        compare newOptimal to Optimal
                        set bestOptimal;
                }
                Optimal = oldOptimal;
                Jobs = oldJobs;
        }
        return bestOptimal;
}
```

Figure III.6: Backtracking Psuedocode.

If the value put on the element being expanded is lower than the bound, then that element will be expanded. If it is not, that portion below the element is pruned off and the algorithm moves on to the next element at the current level. If it reaches a terminal state and the path value is less than the bound value, the bound is set to this new value and the new optimal path is returned. Here, I actually needed to implement my own queue class for determining the value of the bound. This was the only separate class that I need to implement. Java supplied the rest.

Finally, I wanted to integrate animation with user control. This was accomplished by creating buttons that allowed the user to stop, step, and start the algorithm. I created the following function that handles any action given by one of the buttons. There are three boolean variables, suspend and pause and running, that are set when the respective button is pressed. Then, through out the Algorithm Analyzer, there are stop points that check the current state of these variables and adjust the environment (starting, stopping, or stepping) accordingly. Java allows threads to be stopped or resumed, which greatly reduced the implementation time in this portion of the project.

```
BranchANDBound(Jobs, dataStatus, bestValue, bestSol, level)
{
        Setup the valuearray for that node //valuearray->[column,value,bestValue]
        localQueue = Bound(Jobs, valueArray, localQueue);
        while (!localQueue.empty())
        {
                dequeuedArray = localQueue.remove():
                bestValue = dequeuedArray[2];
                Jobs[dequeuedArray[0]] = true;
                bestSol[dequeuedArray[0]] = level;
            if (jobsFull)
            {

                if (bestValue < OptimalValue)
                {
                        OptimalValue = bestValue;
                        OptimalSolution = bestSol;
                }
                Jobs[dequeuedArray[0]] = false;
                return OptimalSolution;
          }
          else
          {
                if (bestValue < OptimalValue)
                {
                        tempLocalQueue = localQueue.saveQueue();
                        OptimalSolution = BranchANDBound(Jobs, dataStatus,
                        bestValue, bestSol, level+1);
                        localQueue.restoreQueue(tempLocalQueue);
                }
          }
          bestSol[dequeuedArray[0]] = -1;
          Jobs[dequeuedArray[0]] = false;
        }
        return OptimalSolution;
}
```

Figure III.7: Branch and Bound Psuedocode.

VI. Conclusion:

The Animated Algorithm Analyzer has been designed and implemented based on student interpretations. It is meant to challenge their normal way of thinking and to allow them to look at the problem in a different way. Keeping this in mind, this project is a useful tool. It is a tool only if the students use it as a resource. The Animated Algorithm Analyzer will be used in The University of Northern Iowa's Computer Science Department. The department will use this tool to teach Algorithm students about optimization problems, such as the Job-Assignment problem, handled by the backtracking and branch and bound algorithms.

Although the development time was an entire semester, the actual coding was approximately one fourth of that. Using Java as the language of choice for this project proved to be quite helpful. The predefined GUI classes quickly became priceless. The programming language by itself is solid and stable and seems to be fairly efficient. But, it is its graphic capabilities that allowed this project to obtain the level that it did reach. If I would have needed to define every graphical class that I used, it is quite possible that I would have not finished even one algorithm. In actuality, the defining of graphical classes for C++ could have been a research project in itself. Java was designed for modern use. By that I mean, the incorporation of internet capabilities and the for-mentioned GUI classes. It was the internet capabilities that persuaded me to select Java as the projects language.

```
Public void handleButton(String buttonName)
{       if(buttonName.equals("Start"))
        {       if(!running)
                {       try
                        {m_backtrack.resume();}
                        catch(InterruptedException e){};
                        suspend = false;
                        pause = false;
                        running = true;
                }
        }
        else if(buttonName.equals("Step"))
        {       if(!pause)
                {       pause = true;
                        suspend = false;
                        running = false;
                }
                else
                {       try
                        {m_backtrack.resume();}
                        catch(InterruptedException e){};
                        suspend = false;
                        running = false;
                }
        }
        else if(buttonName.equals("Stop"))
        {       if(!suspend)
                {       try
                        {m_backtrack.suspend();}
                        catch(InterruptedException e){};
                        suspend = true;
                        pause = false;
                        running = false;
               }
        }
 }
```

Figure III.8: Code to implement stop, start, and step buttons

I feel that the project was a success. Not only did I learn to efficiently program in Java, but also I believe that the tool produced will be useful to individuals in the future. Since this project was started from scratch, I created and followed a project life cycle. I learned a lot about software development in a project life cycle that I can take with me to future opportunities. This project was rewarding and on occasion, frustrating, but the information that I learned while creating it has proven to be invaluable.

References:

[Neapolitan 1996] neapolitan, R., & Nimipour, k.
(1996) Foundations of Algorithms. Lexington, MA: D.C. Health and Company

A SUPERDISPLAY PROJECT

Mark Pavicic
Departments of Computer Science and Electrical Engineering
North Dakota State University, Fargo, ND USA
pavicic@plains.nodak.edu

A superdisplay is a scalable parallel processing system for real-time graphical rendering. We are developing a superdisplay to explore the internal structures of 3D objects such as cells, seeds, tissues, fossils, and rocks. The internal structures are represented as volumetric datasets produced by MRI devices, x-ray CT scanners, confocal microscopes, computer simulations, and the like. Because these datasets may be large, often exceeding a gigabyte in size, visualizing them at interactive rates is difficult. However, by utilizing the otherwise idle computational resources of hundreds of on-campus PCs, our superdisplay will enable its users to "fly through" these datasets in real time.

A Superdisplay Project

Mark Pavicic
Departments of Computer Science and Electrical Engineering
North Dakota State University, Fargo, ND USA
pavicic@plains.nodak.edu

Introduction

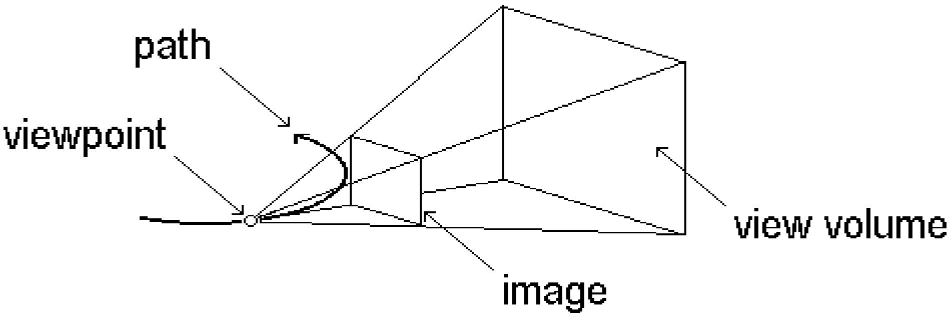
Our objective is to develop a low-cost superdisplay for interactive visualization of volumetric datasets in a classroom setting. This will represent a new application for cluster-based computing and will incorporate a number of innovations in distributed volume rendering.

Volumetric datasets are of interest because they capture information within a volume. They come from a variety of sources including satellites, MR and CT scanners, confocal microscopes, and computer simulations. Furthermore, with the advent of higher-speed networks, it is becoming easier to obtain these datasets from on-line archives, remotely controlled instruments, and distributed computer simulations.

The information in these datasets is most advantageously visualized usingvolume rendering techniques. In addition, it is desirable to be able to "fly through" the data. This means responding to user-directed changes in viewpoint by rapidly generating images of the currently visible portion of the volume. However, because volumetric datasets can be very large, volume rendering at interactive rates is difficult. We propose to solve this problem with a superdisplay.

A superdisplay is a scalable parallel processing system for real-time graphical rendering. The superdisplay we envision will achieve volume rendering at interactive rates by utilizing the computational resources of hundreds of PCs available in on-campus clusters. This will be done while permitting any given PC to be taken over at any moment for traditional personal computing. The output from the superdisplay will be a high-resolution large-area image produced by four projection displays.

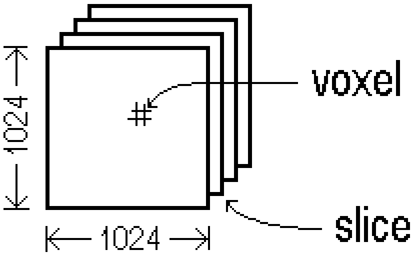
Using the superdisplay, an instructor will be able to take a classroom of students on a visual tour of the normally unseen inner structures of 3D objects such as cells, seeds, tissues, fossils, and rocks. Unlike a movie projector or videotape player, the superdisplay will support interactive exploration. Therefore the instructor will be free to dwell upon or "fly through" whatever portion of the object is of immediate interest.



Flying through a volume. A user of the superdisplay will be able to "fly through" large volumes of data in real time. For smooth motion, the image must be updated at a rate of no less than 10 frames per second (fps).

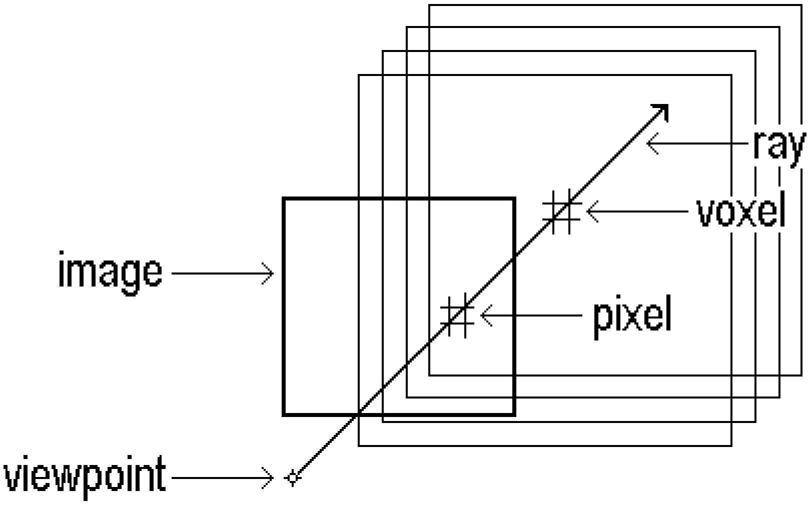
Volume Rendering

Volumetric datasets contain data sampled from numerous points within a volume. The data could be temperature, pressure, density, or whatever. The sample points typically form a regular 3D grid. The region of space nearest to a single sample point is called a volume element or voxel.



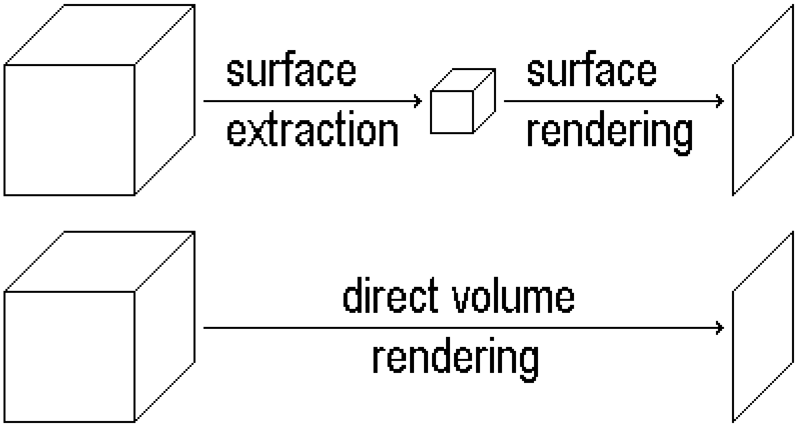
A volumetric dataset. If there is 1 byte/voxel, 1M voxels/slice, and 1K slices/volume, then a single volume occupies 1G bytes of storage.

To visualize a volumetric dataset, an image must be created from the volume. The image is a regular 2D grid of picture elements or pixels. One technique for creating an image is volume rendering. In volume rendering, the color of a pixel is a function of the data that is intersected by a ray or beam. If the image is to be a perspective projection, each ray or beam begins at a common viewpoint, passes through a particular pixel, and intersects with the volume. If the image is to be a parallel projection, then all the rays or beams are in the same direction. There are many choices for the function of the intersected data, such as finding the average value or maximum value and mapping this value to a color.



Volume rendering. An image is created by intersecting the volume with rays or beams.

Another technique is surface rendering. This approach involves two steps. The first step is to extract isosurfaces from the volumetric dataset and approximate them by collections of planar triangles. The second step is to create an image from these surfaces. An advantage of this approach is thatthe second step can be accelerated by exploiting the triangle rendering capabilities of high-performance graphics workstations. The first step must still be done in software, however. Also, there is the disadvantage that extracted surfaces represent only a subset of the information that is within a volume. What is omitted may be of interest.



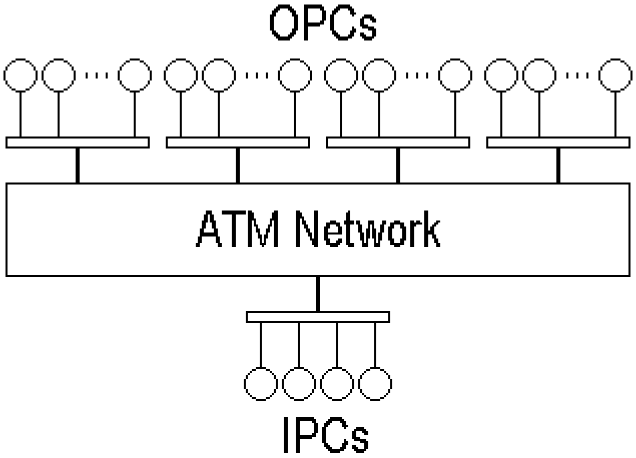
Visualizing volumes. Two approaches.

Some workstations support an alternative approach based on 3D texture mapping. This has many of the features of volume rendering. However, for reasonable performance, the entire dataset must fit within texture memory. Datasets that are larger than texture memory must be handled in pieces, which results in severe performance penalties.

Direct volume rendering enables visualization of information that can not be satisfactorily represented by surfaces. The problem is how to do it quickly. Graphics workstations are not the solution. They are optimized for rendering shaded texture-mapped triangles. Triangles are special because they have a compact representation. This compactness is what makes dedicated hardware pipelines feasible. Volumetric datasets, on the other hand, are not compact. They are orders of magnitude larger than surface representations.

The Superdisplay

To get the storage capacity and processing power to render volumes at interactive rates, our superdisplay is designed to utilize the computational resources of 200+ PCs. These PCs are part of the existing university computing infrastructure and are already connected to the campus network. The challenge is to develop a distributed method of direct volume rendering that achieves real-time performance with non-dedicated resources, and does so without noticeably impacting normal student use.



System hardware. The superdisplay utilizes the existing university computing infrastructure.

The above diagram shows the overall hardware architecture of our superdisplay. The circles are PCs. All the PCs attach to Ethernet hubs. A maximum of 16 PCs are attached to a single hub. The hubs, in turn, are attached to switches within the ATM network. The ATM switches convert between Ethernet packets and ATM cells. Once within the ATM network, data is transferred at 155 Mbps.

The PCs at the top of the diagram are responsible for doing the bulk of the volume rendering tasks. Because the volume resides in object space, these PCs are referred to as object PCs (OPCs). The OPCs are also available for student use. Each OPC has a 10 Mbps Ethernet connection.

The four PCs at the bottom of the diagram are responsible for coordinating superdisplay activities and for receiving, combining, and displaying the image data. Because the image resides in image space, these PCs are referred to as image PCs (IPCs). Unlike the OPCs, they are not available for student use. They are dedicated to the superdisplay. Each IPC has a 100 Mbps fast Ethernet connection.

Each IPC also has a projection display (not pictured). The four projectors are arranged so that each projector displays one quadrant of the image. The result is a large-area display that can be viewed by a gathering of collaborators or students.

Cluster-Based Computing

The idea of utilizing idle cycles of clusters of low-cost systems is not new. Many clusters have been built using PVM (parallel virtual machine), the most popular software package for this purpose. PVM was originally written for Unix systems, but has recently been ported to Win32 systems. Other examples of clusters are Condor (U of Wisconsin at Madison), Symbio (U of Illinois at Urbana-Champaign), and Calypso (New York U and Arizona State U). Condor uses Unix workstations, but Symbio runs on Windows/NT machines and Calypso runs on Windows/NT or Windows/95 machines.

Although the idea of cluster-based computing is not new, the superdisplay is a new application that presents some unique challenges. The usual goal of builders of cluster-based systems is to create an inexpensive virtual supercomputer on which to process long-running C programs. Therefore most implementations are based on Unix workstations that are expected to chip away in parallel at massive computations that take hours or days to complete. Our superdisplay, on the other hand, is based on commonly available PCs running Windows/95. More significantly, it is a real-time system. Instead of hours or days, it must be able to complete its main task, producing an image, in less than 100 ms.

Distributed Volume Rendering

It is presumed that the volumetric datasets will be fetched from their original sources and stored in local servers. When a dataset is to be processed by the superdisplay, it is copied from the servers into the IPCs. The IPCs partition the dataset and create a pool of short duration tasks.

Each OPC is enhanced with a lightweight distributed operating system kernel designed to service a small number of tasks. The kernel is activated after the OPC has been idle for a certain period of time. It is deactivated immediately after any event that may have been caused by a local user. After its kernel has been activated, an OPC acquires a number of tasks from the pool of tasks created and maintained by the IPCs.

Each time an image is needed, the IPCs send messages to the OPCs, giving them additional information they need to perform their tasks. In response to these messages, the OPCs perform their tasks and send the results to the IPCs. The IPCs collect the results and combine them to assemble the image to be displayed. When it is completed, the image is passed to the display projectors.

Complications

Assuming a volume of 1G voxels and an image update rate of 10 frames/sec, the superdisplay must be capable of rendering 10G voxels/sec. A few quick calculations demonstrates that this is going to be difficult.

First, if there are 256 OPCs, each OPC will be responsible for 4M voxels. Assuming 1 byte/voxel and a transfer rate of 8 Mbps (over a 10 Mbps Ethernet link), it will take 4 seconds for an OPC to load its voxels. This is significant if OPCs switch frequently between serving the superdisplay and serving local users.

Next, each time a new image is needed, the OPC must process its 4M voxels and send the results to the IPCs. This must be done in less than 100 ms. Using a 160 MIPS processor, there are only 4 instructions/voxel. That is not enough.

Sending the results takes time. Suppose the 4M voxels are organized as 16 cubes that are 64 voxels on a side. If each cube projects onto an area that is 64 pixels on a side, then, assuming 4 bytes/pixel, the results occupy 256K bytes. At 8 Mbps, it will take 250ms to transfer these results. That is too long.

Finally, the IPCs need to receive and composite all the image data received from the OPCs. Assuming a transfer rate of 80 Mbps (over a 100 Mbps fast Ethernet link), 1M bytes can be transferred in 100ms. But the 256 OPCs need to send a total of about 64M bytes. Not good. And, since there are four IPCs, each must composite 16M bytes of image data. Using a 160 MIPS processor, there is only 1 instruction/pixel. Ugh.

There are additional complications. Contention for the network will add queuing delays. Furthermore, the availability of any particular shared resource is not guaranteed. The OPCs can be switched to a local user at any time, and most of the network is simultaneously in use for other purposes. Only the IPCs and the Ethernet LAN that interconnects them are dedicated to the superdisplay.

Dealing with these problems is what makes this project interesting. A number of ideas will be investigated. These include exploiting spatial and temporal coherence for data compression, interpolating frames from depth layered versions of key frames, clipping away unseen parts of the volume, varying resolution with depth, and dynamically balancing the load. Also, to deal with late or lost results, the IPCs will generate a low resolution version of the image that can be substituted when and where necessary.

Status

Most of the superdisplay hardware is already part of the existing university computing infrastructure. One exception is the 100 Mbps fast Ethernet link, but this can be supplied by our Information Technology Services (ITS) group. The design also calls for four dedicated IPCs and four color projection displays. The IPCs need to have fast processors, large amounts of memory and disk space, and high quality graphics cards. A PC with these characteristics can be purchased for less than $4000. The projection displays are more expensive, but they can be borrowed from ITS for short periods of time.

With the hardware relatively fixed, the bulk of the work to be done is with the software. Most of this is in the formulative stage. It is clear that a brute force approach will not achieve our performance goals. Therefore it will be necessary to prototype the OPC and IPC and to experiment with many possible optimizations.

Conclusions

Volumetric datasets have significant educational potential for those who are equipped to visualize their contents. This potential is growing as high-speed networks make on-line archives more accessible. We propose to solve the equipment problem with a superdisplay. Our superdisplay will be a distributed processing system utilizing hundreds of PCs already existing in clusters at different locations on our university campus. These PCs are used only when they would otherwise be idle. The biggest challenge is in developing the software that can finesse the massive task of rendering large volumetric datasets at interactive rates. If we succeed, we will have shown how to build a powerful classroom tool for entering and exploring the hidden 3D worlds of volumetric datasets.

Internet Research in the Humanities:
Teaching Critical Thinking to First-year Students

Molly Pederson, M.L.S., M.L.A
Ylvisaker Library
Concordia College
fink@cord.edu

David Sprunger, Ph.D.
English Department
Concordia College
sprunger@cord.edu

Student use of the World Wide Web in humanities courses has never been greater. Yet, as students rush enthusiastically to computer labs and Internet searches, faculty response to this new tool has been mixed.

It's easy to find critics of student Internet research. Such Luddites lament, often rightly so, that students search randomly and then settle for non-authoritative resources, failing to think critically about the information they utilize. At the same time, the Internet offers unprecedented research opportunities for students who can now tap into Humanities resources far beyond the scope of most institutional libraries.

Our mission as educators is to shape students into savvy consumers of Internet data. This session will review the attitudes encouraging indiscriminate and uncritical Internet research. It will then describe one specific model for a session designed to make students efficient and intelligent Internet researchers.

Internet Research in the Humanities:
Teaching Critical Thinking to First-year Students

Molly Pederson, M.L.S., M.L.A.
Ylvisaker Library
Concordia College
fink@cord.edu

David Sprunger, Ph.D.
English Department
Concordia College
sprunger@cord.edu

Introduction

Where have all the bookworms gone? Authors Walt Crawford and Michael Gorman assert that while the death of public libraries and the printed word has often been hinted at in the past, both institutions continue today. They are quick to point out that "more than two-thirds of all U.S. households buy books in a typical year, and more than two-thirds of adults also use public libraries" (p. 34). The increased popularity of businesses like Barnes and Noble seem to indicate that reading good literature is in fact still valued by many people, young and old.

However, it is also true that increasingly, electronic information is becoming more and more central to peoples lives. Currently, it is estimated that there are over five million electronic information bases worldwide and that the volume of electronic information is doubling almost every two years (Rawlins, 1996). A 1997 *Wall Street Journal* study concludes that computer sales will constitute over 1.3 percent of total consumer spending, or about 12 times the percentage spent seven years ago (Tapscott, 1998). The study also shows that "while seven years ago consumers were obsessed with acquiring the biggest RAM on the block, now they're more concerned with using their PCs as tools to access the Net" (p. 22).

Increased access to electronic information has greatly impacted educators and, perhaps more dramatically, their students. In 1993, the U. S. National Center for Education Statistics reported that 55.2% of all undergraduate (1st-4th year) college students used computers at school and 32.8% used them regularly at home (*Statistical Abstract of the United States*, 1997). At that time, very few people had access to or even knew what the Internet was. By 1996, Mediamark Research Incorporated reported 43.2% of their survey respondents aged 18-34 had direct Internet access either at home or at work (*Statistical Abstract of the United States*, 1997). It is estimated that by the year 2000, over 40 percent of American households will be directly connected (Tapscott, 1998). These statistics show a dramatic increase in student computer use and online experience.

The influence of such computer technology continues after students graduate. While over a third of all American households have computers, that proportion jumps to well over half for homes with university graduates (Rawlins, 1998). Clearly our students are embracing the Internet and accessing more online information than ever before.

College students find Internet research easy, engaging, and fun. The Internet is exciting and most students are enthusiastic about using it. In fact, some college officials debate whether their students are spending *too much* time on the Internet (DeLoughry, 1996). The term "Internet addiction" describes people with "a psychological dependence on the Internet," and evidence suggests that college students are especially vulnerable to this new obsession (Young, 1998, p. A25). Learning good time management skills can be particularly difficult for first-year college students. However, Internet use is not the *only* area where students must learn to prioritize. Internet addiction is a controversial topic, and while it can be cause for concern, many experts find it difficult to get too upset about it:

> The issue is one of balance. If a child becomes involved for a prolonged period of time in something which is causing disequilibrium in his life, we should be concerned. If she is giving up her sports team, homework is suffering, friends are being neglected-then there may be cause for concern. However, experience shows that compulsive use of the new media is fairly rare and that when it occurs it is usually a temporary problem. (Tapscott, 1998, p. 117).

If college is a place where young adults solidify their work habits and critical thinking skills, surely one of our responsibilities as educators is to help them assess new technologies critically and help them see their value as well as their potential limitations.

The Problems

College faculty are scrambling to adopt information technology into their courses. Many are trying to create courses that are more interactive and collaborative. A quick search of the ERIC (Educational Resources Information Center) database shows that increasingly faculty have developed online course syllabi, class listserves and interactive Web assignments. Neil Rudenstine (1997), president of Harvard University, comments on what appears to be a national trend:

> …we see evidence of rapidly expanding educational use. Whether one considers the number of academic institutions or courses with sites on the Internet, the content and dimensions of these sites, the number of "hits" registered by a given home page, the volume of academically significant materials coming online or the volume of e-mail messages transmitted each day by faculty members and students, it's clear that we are experiencing a phenomenon of rare magnitude. (p. A48)

Yet, students are often not adequately equipped to deal critically with the potential perils of these new technologies.

Students face many challenges when using the Internet. They often search haphazardly for information and when they stumble across it, they use it indiscriminately. This becomes especially problematic because as Bryan T. Sinclair (1997) notes, "Students (now) have greater access to more poor-quality information than ever before" (p. 5). David Rothenberg (1997) goes so far as to suggest that student reliance on inferior Internet research has resulted in "a disturbing decline in both the quality of the writing and the originality of the thoughts expressed" (p. A44). Three major areas were of concern to us as we began thinking about designing a freshman writing class that required students to utilize at least one Internet resource.

First, we wanted students to become more effective Internet searchers. Several things challenge students in this area. Often students use public computer labs that require them to constantly redo their searches. Because these students do not have the luxury of configuring their own personal browsers, their research is less efficient. Also, first-year students often do not have a clear understanding of search engines and why or how they work. Most do not realize there are so many search utilities or that they cover vastly different portions of the Internet. While students may be familiar with one or two of the more commercialized engines, most are unaware that there are specific strategies for searching within them. They are not comfortable guessing site addressees, using truncation symbols, or limiting their searches. Students become particularly anxious if they access a non-working address or revisit a site that has suddenly disappeared.

Secondly, we wanted students to develop better critical thinking skills while working with Internet sources. This can be additionally challenging because first-year students are still developing these skills with printed resources. However, we hoped to keep students from succumbing to what Crawford and Gorman (1995) have termed "technolust." First year students often turn into "Wired People" the instant they trip over a site that seems to cover their paper topic:

> Wired People assume that a periodic table available over the Internet must be right—after all, it is on the Net—even if some of the symbols are wrong and it seems to be missing a couple of dozen elements. Wired People send out questions and assume that the answers they get must be correct, -- they have been sanctified by the Net—but really have no time to deal with the answers because they are too busy with something new on the Net. (p. 41)

To combat such "technolust" we encouraged students to be critical about their sites. Evaluation and assessment was an issue for both printed and electronic resources. We wanted students to assess sites in these key areas:

- Who wrote the material at this site? Why is she or he a credible source? What else do we know (or can we learn) about that person?
- When was the site created? When was it last updated? The pressure to keep a site current is much stronger in the commercial world than the academic, and many–if not most–academic sites are updated only infrequently, but we wanted students to think about dates and to consider if a site's dated material affected its relevance to their projects.
- Is the material at this site primary or secondary? Students in the humanities are much more likely to locate secondary research, and that's exactly the sort of material that most cries out for careful critical evaluation.
- What is the basis for the material's authority? How was it gathered? What sorts of sources did the author consult? Are those sources identified?
- From what domain does the site originate? Some students will automatically trust an .edu site and be suspicious of .com or .org sites, but these labels should raise flags for examination rather than evoke automatic judgments.
- What seems to the site's primary reason for existence? Why has someone gone to the trouble of collecting, coding, and posting the information? What agenda does the site creator have? Once the student recognizes any bias, does that mean the site is not relevant to the research project?

Finally, we hoped to keep students from experiencing too great a sense of information overload. We did not want them to feel as though they had to find *everything* on their topic through the Internet. We also did not expect them to become Internet searching experts. We knew that trying to teach students too much is a particularly dangerous trap for instructors who have a lot of experience working with the Internet. Gertrude Himmelfarb (1996) touches on this danger when she writes, "What is important in the history of ideas is not retrieving and recombining material, but understanding it. And that requires a different relation to the text, a different tempo of reading and study" (p. A56). We did not want to overwhelm students with the Internet. We hoped our students would focus on the content of a few Web pages and not become caught up in blinking text, glamorous pictures and animated gifs. We did not want them to simply "recombine" information, but to absorb it. To accomplish this task, we decided to keep our Internet session simple.

The Ideal Solution (or Living in a Dream World)

In the best of all worlds, students would be savvy searchers and critical thinkers; web sites would be accurate, clearly indicate their biases, and be perpetually updated. A fully networked computer lab with projection capabilities would be available for class instruction at the instructor's whim, and several class sessions could be devoted to the topic. A professional librarian might also be present for consultation and guidance during each of these sessions. The theoretical discussion would be reinforced with practice, and the practice would lead to discussion and further insight because it was connected to the students' own lives.

The Reality

The reality is that we needed to accomplish our plan in a limited amount of time with the resources we had. We were fortunate enough to have access to a fully networked lab with projection capabilities. The lab features twenty-four machines running Netscape Navigator Gold. Because the class had only nineteen students, each was able to sit at his or her own terminal and could gain valuable hands-on experience. The following is an outline of our single 70 minute session:

- ❖ Students learn how to use bookmarks and save them to a disk (15 minutes). The instructor asked students to turn in these disks with their bookmarks as part of the final paper project. The disks were also to be used during scheduled conferences with their professor.
    - Practice Activity 1: Students point their browsers to a search utility (www.findspot.com) and save that location to their bookmark file in a folder called "search tools." [Appendix 1: Using Netscape Bookmarks; Appendix 2: Internet Research Resources]

- ❖ Students then receive basic instruction about Internet searching tools (20 minutes). (See handouts.)
    - Practice Activity 2: Students use bookmark from Activity 1 to search for several sites relevant to their individual topics. They save these additional sites to their bookmark file in a folder titled "research links."

- ❖ Finally, after students locate one or two web sites that apply to their research, they learn to evaluate several key features of their sites (35 minutes).
    - Practice Activity 3: A. Students pair up and apply the evaluation criteria to one of the sites each bookmarked in Activity 2. [Appendix 3: Web Site Evaluation] B. Students then take turns showing their pages on the projector and telling classmates what strengths and weaknesses they note. The professor and librarian supplement this discussion by highlighting additional areas the students left uncovered.

Reinforcement of this lesson takes place with several out of class assignments requiring minimum feedback from the teacher. Some examples include these:

- using bookmarks outside of the lab, in class, and in individual conferences
- "show and tell" using web pages saved outside of class
- Search Engine Comparison [Appendix 4: Search Engine Comparison]
- Web Site evaluation [Appendix 5: Web Site Analysis]

The final reaction to this session is positive. Students report that they continue to use bookmarks, often beyond the minimum requirements of this class. One student reported collecting over 400 bookmarks in only two days (perhaps another victim of "technolust")! The Search Engine Comparison and Web Page Evaluation assignments found students raising issues and asking questions that showed a heightened awareness of the critical areas we had stressed in the workshop. And, in the place that mattered most, the research essay, student use of the web material was on par with their use of printed sources.

Appendix

We developed the following handouts for our sessions on WWW searching and critical thinking. They may be freely duplicated and adapted as long as our attribution is retained.

1. Using Netscape Bookmarks
2. Internet Research Resources
3. Web Site Evaluation
4. Search Engine Comparison
5. Web Site Analysis

Appendix 1: Bookmarks

USING NETSCAPE BOOKMARKS

An important skill in conducting web research is being able to save useful locations in a bookmark file. That way you don=t have to search for them again.

Adding, Removing, and Organizing Bookmarks

1. When you=re on a page that you want to save, click on the Abookmark quick file@ bar and select Aadd bookmark.@
2. To see your bookmarks, either select them from the Abookmark quick file@ bar or use the shortcut A^b@
3. To remove a bookmark that you don=t need any more, highlight the shortcut and press the *delete* key.
4. To organize bookmarks, create some folders and assign them logical names (e.g., Aresearch links@). Then, once you=ve added a bookmark, you can Adrag and drop@ the link into the appropriate file.

Saving the Bookmark File

1. After you=ve created your bookmark file for the first time, save the file to your floppy disk by using the command *File/Save as*. Switch your directory to drive a:.
2. It=s least confusing to leave the file name Abookmark.html.@
3. Don=t forget to take your disk with you!

Using the Bookmark File

1. Once you=re running Netscape, open the bookmark page, either by selecting it from the Abookmark quick file@ bar or by using the shortcut A^b@
2. When using machines in the lab, clear the existing bookmarks, by first using the command *Edit/Select all* and then pressing the *delete* key before you create your own bookmark file or import an existing bookmark file.
3. Use the command *File/Import* to call up your bookmark page.
4. When you=re done browsing, use the command *File/Save* to update the bookmark file on your floppy.
5. If you only want to use (or show off) your file and know that you won=t be adding to it, try the command *File/Open page* and enter the location and name of your file (e.g., Aa:bookmark.html@).

Appendix 2: Internet Research Resources

Webliography: Simple Sites for Getting Started

Netiquette:

The Core Rules of Netiquette, exerpts from the book *Netiquette* by Virginia Shea. Pub. Albion Cybercasting.
http://www.albion.com/netiquette/corerules.html

Search Utilities:

Findspot by Digital Tools and Designs, Inc., 1996, 1997.
http://www.findspot.com/
"*Findspot* is a gateway to Internet search tools that people can use to find information. Its purpose is to combine access to these tools with example based help so that users can construct better searches, leading to more useful retrieval."

Search Engine Showdown by Greg R. Notess, Montana State University-Bozeman Library.
http://imt.net/~notess/compeng.html
"This site summarizes, reviews, and compares the search features and database scope of the Internet search engines and finding aids."

"Search Engines Facts and Fun." (Search Engine Watch.) Ed. Danny Sullivan. Mecklermedia, 1996-1998.
http://searchenginewatch.com/facts/index.html
"This section of Search Engine Watch provides background about search engines, tips on how to use them better, some history and even a game to test your knowledge."

Web Searching--Faster, Faster! by Michael Cahlin and Harry McCracken. (March 1998 Issue of *PC World*.)
http://www.pcworld.com/software/internet_www/articles/mar98/1603p156.html
"Follow the links to find the best sites, tips, and techniques for combing the Net."

Evaluation:

Evaluating World Wide Web Information by Ann Scholz, Rutgers University Libraries.
http://crab.rutgers.edu/~scholzcr/eval.html

The Good, the Bad and the Ugly or, Why It's a Good Idea to Evaluate Web Sources by Susan E. Beck, Instruction Coordinator, New Mexico State University Library.
http://lib.nmsu.edu/staff/susabeck/eval.html

Ten C's for Evaluating Internet Resources by Betsy Richmond, McIntyre Library, University of Wisconsin-Eau Claire.
http://www.uwec.edu/Admin/Library/10cs.html

Thinking Critically about World Wide Web Resources by Esther Grassian, UCLA College Library.
http://www.library.ucla.edu/libraries/college/instruct/critical.htm

Sprunger/Pederson, 1998

Appendix 3: Web Site Evaluation

URL: ______________________________________

What can you infer by looking at the site's address? (Pay special attention to the server and domain.)

When was this site created? When was it last updated?

What do you know about the person who created the site?

What is the site's main purpose? (What motivated the author to create this site? Is there any obvious bias?)

Is the information at this site primary or secondary?

How could you verify the site's information?

What about this site is superior to information available in the library?

Appendix 4: Search Engine Comparison

WWW Search Engine Comparison

For this assignment, you=ll conduct your own version of the Asearch engine shootout@ and analyze the results. Start by going to *www.findspot.com* and using one of the Aweb meta search utilities.@ Search for some information related to your research topic, **using more than one term and use some sort of Boolean operators**.

0.a. Which meta-search utility did you use?

0.b. What terms and operators did you enter?

After your search is completed, you may need to repeat the search in order to answer these questions. Select two of the search utilities within the meta-search and compare the results.

0.c Identify the two search engines you=ll compare.

1. How many hits did you find in each engine?

2-3. Describe the main differences between the way each searches for information.

4-5. When you search for multiple terms in each engine, how do you keep terms together? For example, if you were searching for *Concordia College*, how can you ensure that you find both terms together rather than everything with AConcordia@ and everything with ACollege@?

6-8. What are some of the most important criteria for evaluating Internet sources? Identify and explain the <u>three</u> that matter most to you.

9-10. Practical:

For the practical portion of the exercise, be prepared to display your computer search savvy in my office. Bring your disk with 1) your organized bookmark file; 2) at least one page of information saved to a file. See the class web site for other skills you may be asked to demonstrate.

Appendix 5: Web Site Analysis

Web Site Analysis

This assignment asks you to provide a detailed formal evaluation of an Internet source. Your goal is not merely to list information, but to *discuss* the information, showing that you're aware of the factors that contribute to a source's credibility.

Include these points:

1. Letter-perfect MLA citation for the site.

2. Describe the kind of information at the site. Is it an essay, a press release, results of a study, or what?
   Try to sum up the site's purpose in a single sentence.

3. Assess the site's credibility.
   a. What do you know about the author or publisher of the web page? What makes the publisher an authority on the topic? Do you learn any institutional affiliation or other credentials? Can you infer anything from the address of the site?
   b. What backs up the material published at this site? Why should you believe the facts, opinions, or so forth?
   c. Do you learn anything about the date of the site? When was it created? Last updated? Is the date too close or too far removed from the topic? Is it too close or too far removed from now to be relevant?
   d. What's the apparent purpose of the web site? To provide factual material, to promote a particular viewpoint? Is there a corporate sponsor? Is there a commercial interest? Any other bias?

4. React to the source. Why would you use this source instead of a printed source from the library? Is it superior (or inferior) in some ways? How might it be of value to your research project? (This section is your opportunity to demonstrate that you understand the limitations and advantages of Internet materials.)

5. Somewhere in your response, include one direct quotation from the source, properly *integrated* and *documented*.

Miscellaneous Points:

— Print out and submit the site that you're evaluating. If you're using only part of a document, cut the relevant section and paste it into Word before printing.

— When you use quotations from your source, they must be <u>integrated</u> with your writing and documented with MLA parenthetical citations. Remember that MLA does not include any page, screen, or paragraph reference for Internet sources.

— The sources you choose to evaluate must be <u>significant</u> for your topic. Don't waste time by evaluating a site that summarizes material readily available elsewhere. If you are not sure a source will count for this option, check with me ahead of time.

Web Site Analysis
Sprunger/Pederson, 1998

References

Crawford, W. & Gorman, M. (1995). *Future libraries: Dreams, madness and reality*. Chicago, IL: American Library Association.

DeLoughry, T. J. (1996, March 1). Snared by the Internet. *The Chronicle of Higher Education,* A25+.

Himmelfarb, G. (1996, Nov. 1). A neo-Luddite reflects on the Internet. *The Chronicle of Higher Education,* A56.

Rawlins, G. J. E (1996). *Moths to the flame*. Cambridge, MA: MIT Press.

Rothenberg, D. (1997, Aug. 15). How the web destroys the quality of students' research papers. *The Chronicle of Higher Education*, A44.

Rudenstine, N. L. (1997, Feb. 21). The Internet and education: A close fit. *The Chronicle of Higher Education*, A48.

Statistical Abstract of the United States (1997). Washington, D.C.: U.S. Dept. of Commerce, Bureau of the Census.

Sinclair, B. T. (1997, March). Teaching students to be critical thinkers on the web. *The Teaching Professor*, 5.

Tapscott, D. (1998). *Growing up digital*. New York: McGraw-Hill.

Young, J.R. (1998, Feb. 6). Students are unusually vulnerable to Internet addiction, article says. *The Chronicle of Higher Education*, A25.

SMILEY: A WEB-BASED REMOTE SENSING DATA MINING SYSTEM[1]

Dr. William Perrizo, Longjun Chen, Dennis Amundson ,Shuxue Li
Computer Science Department
North Dakota State University
Fargo, ND 58105
{perrizo, lchen, amundson}@plains.nodak.edu
shli@prairie.nodak.edu

ABSTRACT

Remote sensing imagery (RSI) data will be used by potentially millions of users, provided that there exists a good way to access and manipulate that data. Currently, the most common way to manipulate and analyze RSI data is through a GIS system. Usually this method of the GIS distribution (both the software and the data) is costly and slow. Furthermore, the cost of owning RSI data is very high. These problems may limit the application and usage of remote sensing data.

In this paper, we describe a novel architecture to access, distribute, and analyze the massive amount of RSI data in order to address the problems described above. This architecture utilizes the most recent Internet technology and provides a distributed multi-tier client/server architecture for accessing and analyzing RSI data. SMILEY, a WWW-based remote sensing imagery data mining system, provides a general interface to access, display, and manipulate various sets of satellite imagery data. Users, through any common WWW browser, can use virtually any computer platform at any location to do RSI data analyzing. The distributed server architecture of SMILEY is scaleable so it can easily be upgraded to handle an increasing workload, thus achieving a high degree of reliability.

[1] Partially supported by grants, NSF No. OSR-9553368 and DARPA No. DAAH04-96-1-0329.

INTRODUCTION

The Earth Observing System (EOS) [EOS 93] is one of NASA's ongoing space- and ground-based measurement systems to provide sustained observations of the earth. The EOS consists of a series of polar-orbiting and low-inclination satellites. Each satellite bears several sensors, for long-term global observations of the land surface, biosphere, solid Earth, atmosphere, polar ice, and oceans. The Earth Observing System Data and Information System (EOSDIS) manages the data sets collected from NASA's earth science research satellites and field measurement programs [Barkstrom 91]. EOSDIS data is only one type of remote sensing imagery (RSI) data. Other RSI data sets include data obtained from other sources, such as radar data and remote digital photography.

Initially, it was assumed that EOSDIS data would be accessed by a few hundred primary EOS earth science investigators and approximately 10,000 additional researchers to carry out basic global-change research [Vetter 95]. Since EOSDIS data belongs to the public, non-research users such as educators, agribusiness marketers, community planners, transportation users, and others who want to access it should be able to do so. Thus, EOSDIS data could potentially be used by millions of individuals, if a good method existed to access and manipulate that data.

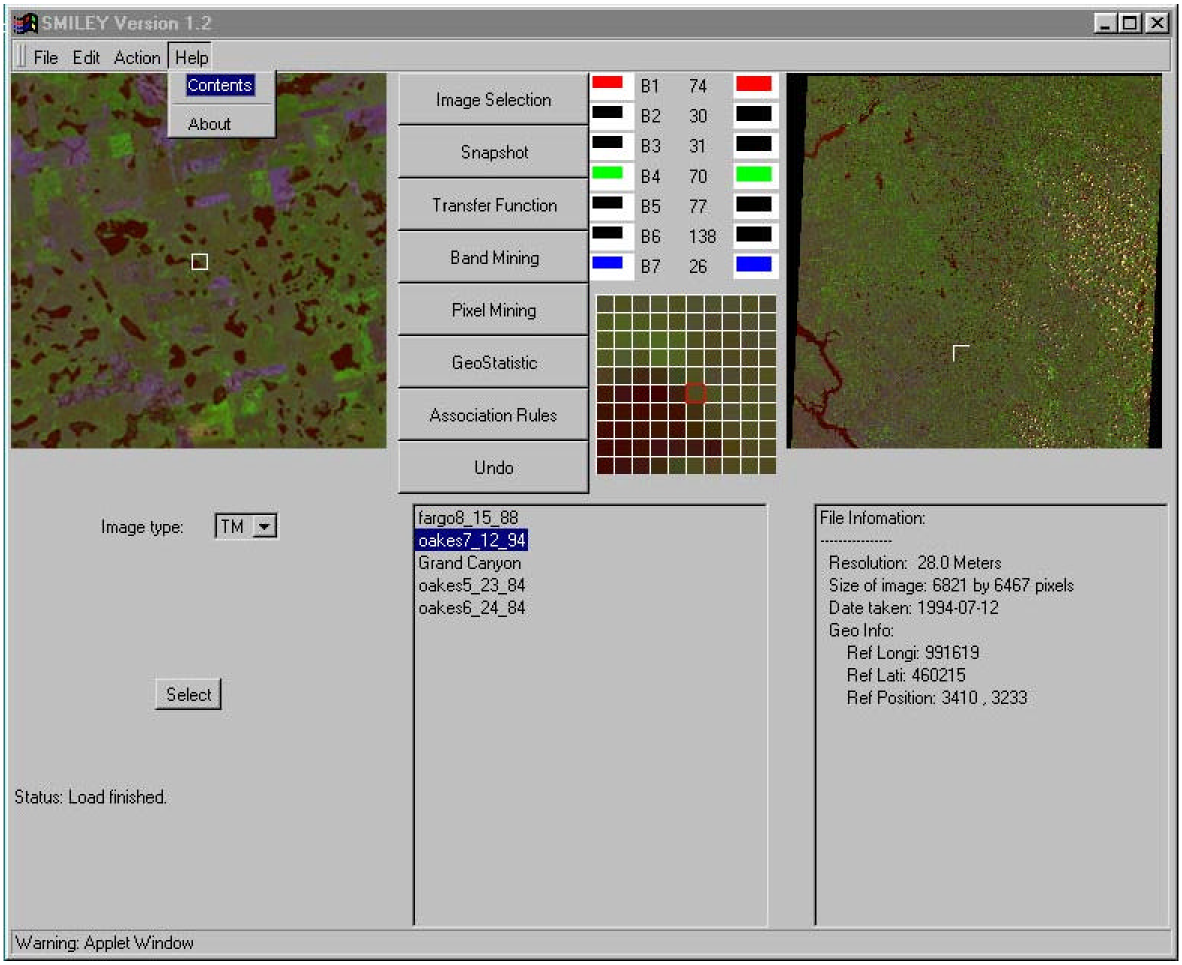
Currently, the most common way to manipulate and analyze EOSDIS data is through a commercial geographic information system (GIS) system. A typical GIS system is a standalone software package which the user purchases for geographical data manipulation. The data sets, which the user wants to use, may be obtained either directly with the purchased GIS system used or bought through third parties. These methods of distributing the GIS software and data are costly and slow. These problems may limit the application and usage of remote sensing data.

In this paper, we provide a novel architecture for accessing, distributing, and analyzing the massive amounts of remote sensing imagery (RSI) data to address the problems described above. This architecture utilizes the most recent Internet technology and make use of a distributed multi-tier client/server architecture for flexible accessing and analyzing of RSI data.

The World Wide Web (WWW) is an attractive, easy to use vehicle that makes data available for heterogeneous clients. At the present time, WWW browsers are available for all major operating systems. Web browsers provide a good platform for one to use and manipulate RSI data retrieved throuh the WWW. *Signature Mining & Interface Language for EOSDIS, Yet-another* (SMILEY) is a powerful online imagery analyzer and viewer. [Figure 1] shows a typical display of SMILEY.

SMILEY provides a general interface using Internet Web-based technology to access, display, and manipulate various sets of RSI data. Eventually, any computer platform from any location on the Internet may be used to do imagery data analyzing. SMILEY uses the client/server model to fully utilize the user's client machine's processing power for achieving quick respond times. Users can also do data mining operations based on individual pixel values in the image or apply band-oriented functions. They can also use many predefined filters (transfer functions) to process the image being analyzed and even define their own filter functions.

This paper analyses the structure and functionality of SMILEY. In [Background], we provide some background information about SMILEY. The structure of SMILEY version 1.2 is described in [Smiley Architecture]. The server and client data mining functions that are provided in the current version of SMILEY are discussed in [Smiley Server and Client Functionalities]. The [Conclusion] provides some suggested future enhancements for SMILEY.



🟥	B1	74	🟥
■	B2	30	■
■	B3	31	■
🟩	B4	70	🟩
■	B5	77	■
■	B6	138	■
🟦	B7	26	🟦

Image type: TM

fargo8_15_88
oakes7_12_94
Grand Canyon
oakes5_23_84
oakes6_24_84

File Infomation:

Resolution: 28.0 Meters
Size of image: 6821 by 6467 pixels
Date taken: 1994-07-12
Geo Info:
Ref Longi: 991619
Ref Lati: 460215
Ref Position: 3410 , 3233

Status: Load finished.

Warning: Applet Window

Figure 1: SMILEY screen snapshot

BACKGROUND

Introduction to Remote Sensing Imagery

Of all data used in spatial data analyzing, some of the most important are those that are obtained by remote sensing. Through the use of satellites, we now have a continuing program of data acquisition for the entire world with time frames ranging anywhere from a couple of weeks to a matter of hours. Very importantly, we also have access to these remotely sensed images in digital form, allowing for rapid integration of the results of remote sensing analysis to spatial data analyzing tools like SMILEY.

Remote sensing can be defined as any process that gathers information about an object, area, or phenomenon without actually being in contact with it. Given this rather general definition, the term remote sensing has come to be associated more specifically with the gauging of interactions between earth surface materials and electromagnetic (EM) energy.

Sensors can be divided into two broad groups–passive and active. *Passive sensors* measure ambient levels of existing sources of energy, while active ones provide their own sources of energy. The majority of remote sensing is done with passive sensors, for which the sun is the major energy source. An example of this type of sensing is airborne photographs that capture energy in the visible spectrum. While aerial photography is still a major form of remote sensing, newer solid technologies have extended capabilities for viewing in the visible and near-infrared wavelengths. By contrast, *active sensors* provide their own sources of energy. A familiar form of this type of sensing is flash photography. However, in environmental applications, a good example of this type of sensing is radar.

When EM energy strikes a material, three types of interaction can follow: reflection, absorption, and/or transmission. For remote sensing, the main concern is with the reflected portion of energy, since that is what is usually returned to the sensor system. The exact amount of reflected energy will vary, and will depend upon the nature of the material and where in the EM spectrum our measurement is being taken. If we look at the nature of this reflected component over a varied range of wavelengths, we can characterize the measurement as a spectral response pattern, which is sometimes called a *signature*.

A signature is a description of the degree of which EM energy is reflected in different regions of the spectrum. Finding distinctive spectrum response patterns is key to most procedures for dealing with computer-assisted interpretation of remotely sensed imagery. This task is non-trivial. Rather, the analyst must interpret the right combination of spectral bands along with the time of year at which distinctive patterns can be found for each of the information classes of interest. A good tool is needed to help correctly interpret the RSI data and identify signatures representing various phenomena.

WWW: an Overview

As the popularity of the Internet increases, people are becoming more aware of its potential. The World Wide Web (WWW, or the web) is a product of the continuous search for innovative ways of sharing information resources among heterogeneous machines and platforms. The implementation of the Web follows a standard client-server model. In this model, a user on a client machine executes a program (i.e., a web browser) to connect to a remote server machine (the web server) where a document store is located. A typical Web transaction takes place when the client submits a request to access a particular document within the store, which the server usually complies with by sending back a copy of the requested document. Most documents on the web are defined using a document markup language called Hypertext Markup Language (HTML), which allows one to easily add hyper-links to the text, which link to other HTML documents, as well as to other types of media (such as graphic images). Web browsers are GUI-like in nature. Mainly because of these attributes, and its ease of use, the WWW has become very popular among Internet users. At the present time, Web browsers are implemented on most major operating systems.

The typical transaction model of the Web is shown in [Figure 2]. The World Wide Web is a client/server application. The server hosts the HTML content and waits for a request by a client. When the server receives this request, it will send the requested document back to the client. The client has a WWW browser that is used to request, receive, and display the HTML document. Most HTML documents consist of static text and/or image files that can be downloaded from the server when requested by the client. But with the help of Common Gateway Interface (CGI) scripts, a mechanism that allows WWW servers to execute special-purpose programs to handle specific requests, HTML documents can now contain some dynamic information based on the client’ s special needs.

While HTML and CGI were intended to add simple user interactivity to WWW server sites, they are also an important means of providing the user a simple GUI-like interface to a distributed application without having to install additional software into the user’ s client machine. However, these tools are not enough to handle complicated data-mining tasks. The limited layout control for the GUI and the lack of continuous execution state on either the client or server sides of the SMILEY interaction indicate a need for a more sophisticated set of tools.

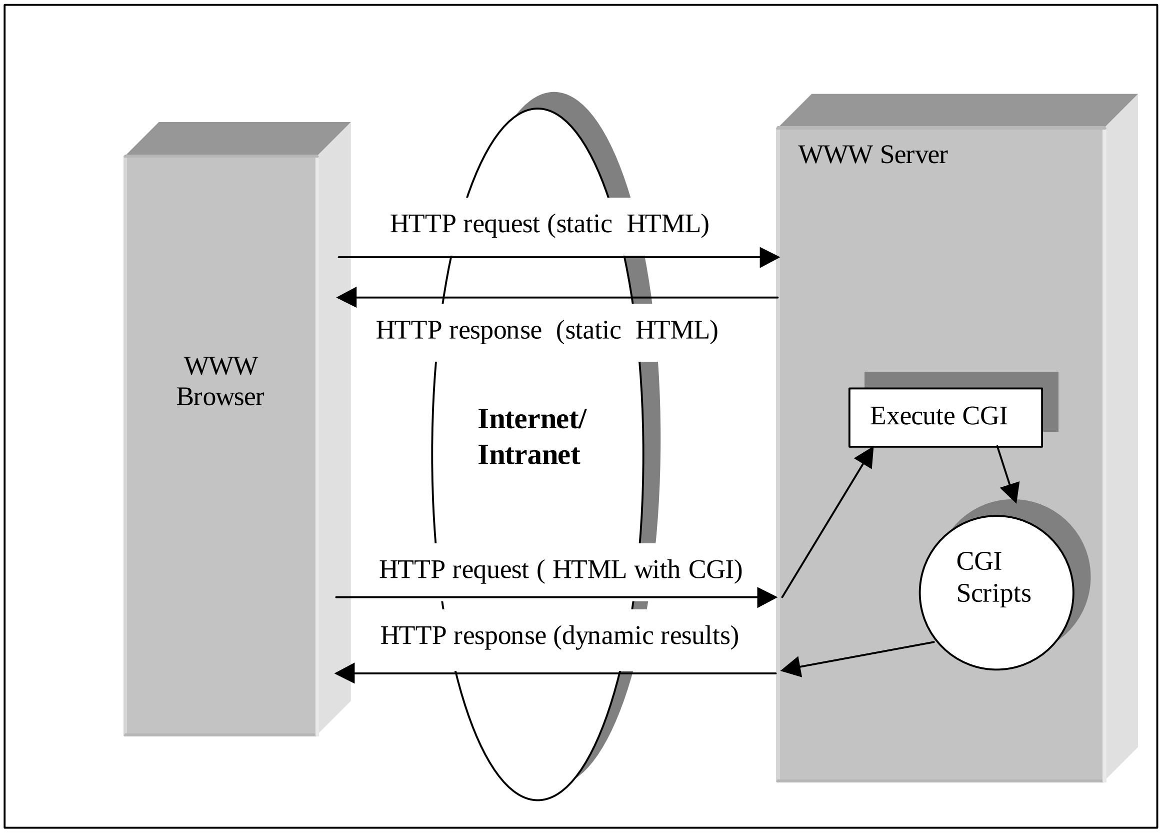

WWW Browser
WWW Server
HTTP request (static HTML)
HTTP response (static HTML)
Internet/ Intranet
Execute CGI
HTTP request (HTML with CGI)
HTTP response (dynamic results)
CGI Scripts

Figure 2: Typical WWW transaction model

Java: a Brief Overview

Java is a new object-oriented programming language. Originally developed by Sun Microsystems for hand-held electronic devices, Java was engineered to be small and platform-independent, making it well suited to apply to the Internet environment. Java itself is object-oriented, implemented especially for network computing applications. Java has many useful features. An important one is that Java source code can be compiled into a machine-independent format, which consists of a set of virtual machine instructions and symbolic data, commonly referred to as *byte-code*. On execution, a Java interpreter acts as the virtual machine and will interpret this byte-code. Most sophisticated web browsers have Java interpreters embedded within them. Thus, Java byte-code can be downloaded along with a refering HTML file to a web client machine and then be executed locally on the client' s machine.

Application domain of SMILEY

SMILEY has a very wide application domain. It can be used in various scientific fields, such as precision agriculture [Brach 74], environmental protection, volcanic activity analysis, earth surface analysis, earth science education, etc. Various kinds of data mining can involve tools and techniques such as neural networks, genetic algorithms, expert systems, database filters, etc.

- Precision Agriculture
  A large store of land-process data is collected from the LANDSAT series of earth observing satellites (part of EOS), and is currently stored at the USGS EROS Data Center located in South Dakota. Agriculture scientists and farmers can use SMILEY to process data, track site-specific farms, and

monitor crop yields. SMILEY is gaining visibility within many precision agriculture and agri-business concerns.

- Education
  The SMILEY analyzer and satellite imagery viewer will be used in several teaching and learning projects. The Upper Midwest Aerospace Consortium (a five-state effort to provide leadership in the areas of Earth System Science) will be using it in its K-12 education component. In that component, school children and young adults will be able to view and manipulate images showing various places of interest. They will be able to do a wide range of interesting projects with the tool. Some of these include: viewing their farm for vegetation quality during the growing season, searching for grasshopper infestations, watching the spring surface water situation for flooding, monitoring wetlands drainage, as well as a number of other projects. These possibilities are limited only by the imagination of the users.

SMILEY ARCHITECTURE

SMILEY which stands for *Signature Miner & interface Language for Earth-data, Yet-another* and is an online tool used for viewing and analyzing remote sensing imagery data. Using current Internet distributed processing techniques (such as Java and the WWW), SMILEY is being designed for the use of many types of remote users, who could be operating anywhere in the world.

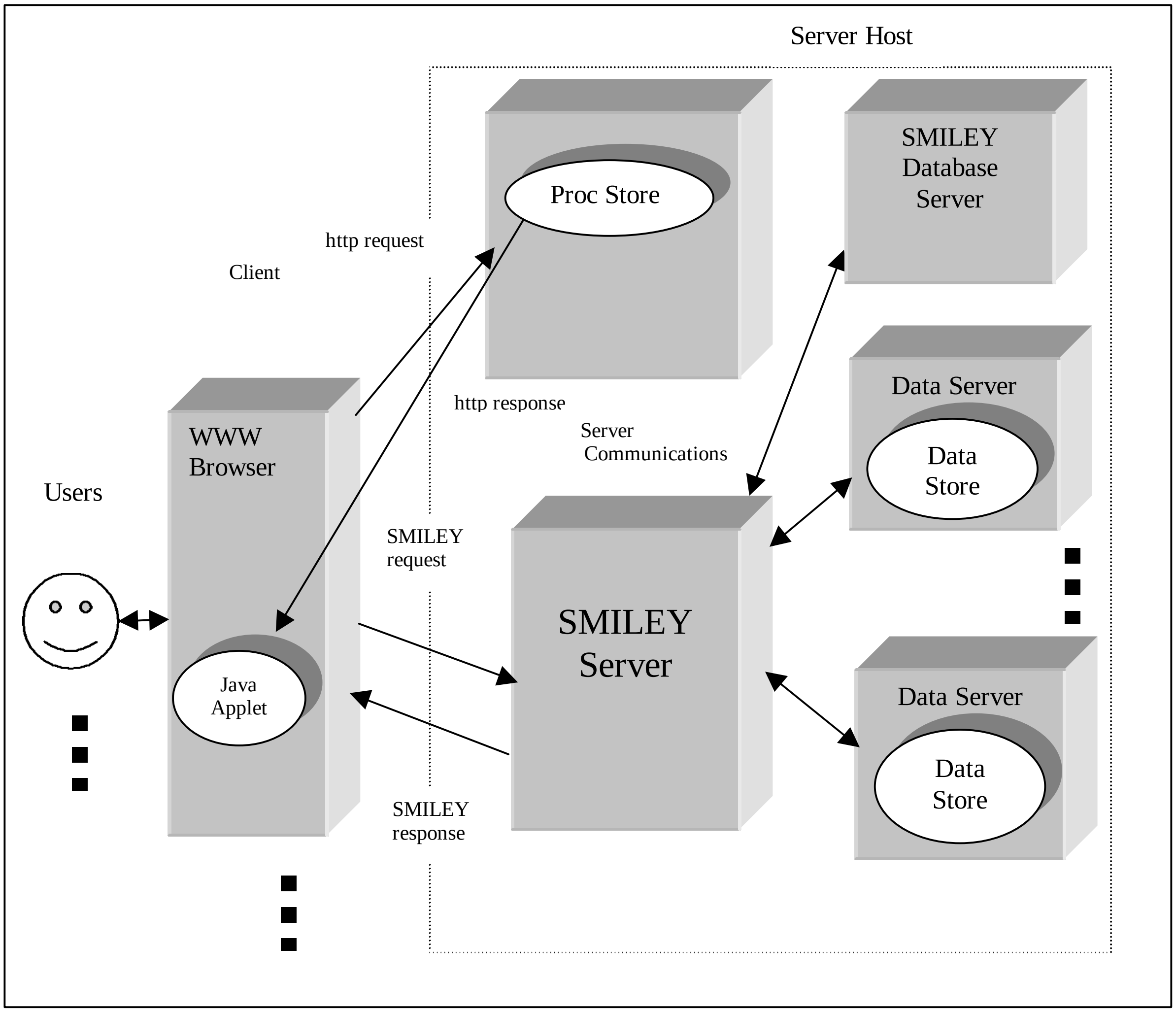


Figure 3: SMILEY (V1.2) System Architecture

There are two main classes of components which constitute SMILEY. One set of components are passive in nature (stores of data and procedures), while the others are active.

The two passive components are:

- Data sets of satellite imagery (data store).
  The data collection of satellite images can be stored either locally (with respect to other components) or remotely (being accessible through a broadband network). Because the volume of the data store can be so massive in nature (one uncompressed band of a TM scene requires 44MB of storage, and normally one TM scene consists of 7 bands), it is likely that most data will be stored away from the active components. Ideally, data is stored in a hierarchical fashion, in descending order of popularity. This allows hot (i.e. heavily used) data to be stored relatively close to the active components for quick access. Data that is relatively cool in nature would probably be stored at a distant location, such as an archive host accessible via a high-speed network or a near-line jukebox of persistent storage.
- Set of stored procedures (proc store).
  The main functionality of SMILEY is implemented as a set of procedures written in Java. These, along with other additional miscellaneous documents (like on-line help pages) and procedures, are stored in the proc store.

The active components of SMILEY are:

- Remote WWW browser (or web browser).
  It is expected that the remote user will have on his/her local system a web browser that is capable of executing the applets composing SMILEY (from the proc store). At the present time, most graphical web browsers have an embedded Java interpreter to allow seamless integration of applet execution with normal web browsing.
- WWW server (or Web server).
  A WWW server is mainly used for transferring hypertext documents (written in HTML) from a central store to a WWW browser. The server may also be used to transfer other types of documents (such as graphic files) and for execution of procedures (typically called CGI programs) to accomplish simple tasks (such as authentication, or dynamic file creation). SMILEY uses the web server for transferring data from the procedure store to the web browser at the remote site. When a remote user invokes SMILEY, the applets that compose its functionality will be downloaded to the browser via the web server.
- SMILEY server
  The purpose of the SMILEY server is to catalog the items composing the data store and to transfer to remotely executing applets the necessary remote sensing data requested by them. Many times, items within the data store are not formatted as the applets would like them. As a result, the SMILEY server needs additional functionality to perform simple graphics-oriented tasks (such as clipping and rotating of images, filtering out one or more bands of the original images, etc.) The complexity of this server varies directly with the complexity of the data store.

SMILEY SERVER AND CLIENT FUNCTIONALITIES

SMILEY is programmed using Java for its primary language. As discussed in [Java], Java is an object-oriented language. It is platform-independent and designed for purposes of network computing.

A diagram representing the general functionality of SMILEY is shown in [Figure 4]. The functionality can be divided into two parts: client-side processes and server-side processes. The client' s main functional purpose is to do image data mining processing and result visualization; the server' s main purpose is to process user registration information and to process image data requests initiated from client processes. At present, most client machines have very powerful processing abilities. To take advantage of this fact, SMILEY distributes its data mining calculation and visualization processing onto the client, in order to achieve faster response times, and to decrease the load on the server.

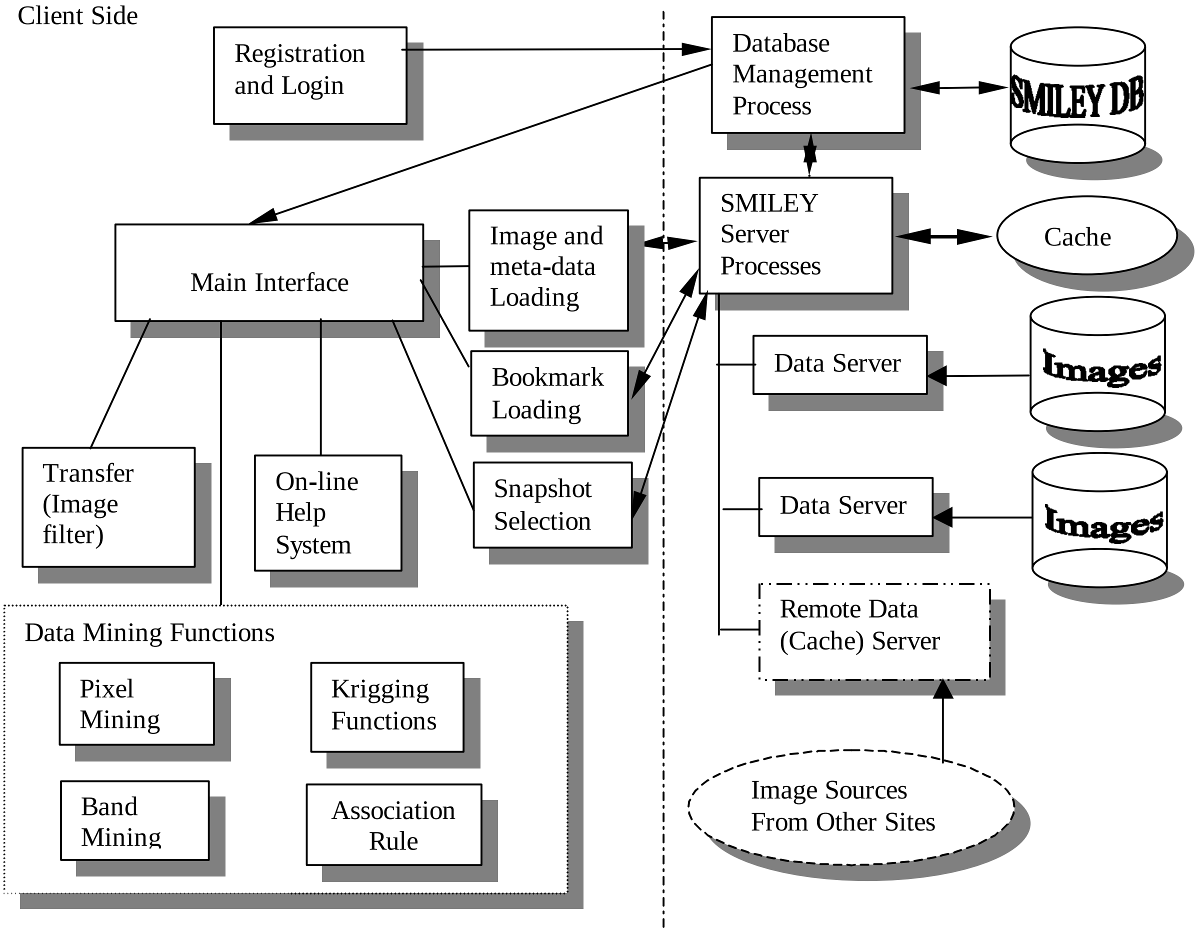


Figure 4: SMILEY (V1.2) software structure

SMILEY Server Processes

The SMILEY servers are responsible for accepting client requests, maintaining image data indices, retrieving and pre-formatting image data, and sending image data back to the requesting client. The functionality of the SMILEY server can be divided into three parts: request handling and dispatching, data indices managing, and data retrieving and pre-formatting. These functions do not necessarily need to be implemented on the same machine. Dividing up the server functionality gives more flexibility for the server to cope with an increasing workload.

SMILEY Server

The SMILEY server uses the concurrent server technique; it can serve an increased workload with better efficiency than one single server can. The SMILEY server acts as a task dispatcher, as shown in [Figure 5].

When the SMILEY server receives a client request, it will check the request against its internal dispatch table. When it finds the appropriate data server or other server (such as index server for browsing request) to answer this request, it will dispatch the task to this server and make itself available to handle next request.

SMILEY Database Management System

The SMILEY DBMS manages a database to store image meta-data, user information and yield map information, etc. The SMILEY DBMS acts as an information center for the SMILEY Server and processes all database operations that originate from the SMILEY Server and the CGI programs. The database schema is shown in [Figure 6].

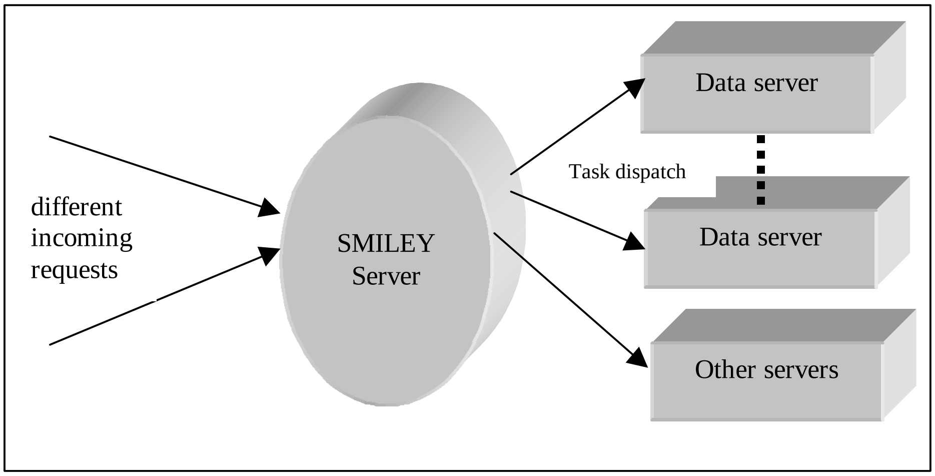


Figure 5: SMILEY server task dispatch diagram

Using a DBMS to manage all the meta-data of the SMILEY Server is desired, although it does add a little overhead to the processing time. When using this model to organize the data sets, management becomes much easier and more flexible when users search for and browse images.

SMILEY Data Server

One SMILEY data server is designed to serve one specific imagery data type. Because each remote sensing data has a different format and structure, and furthermore, each data set in the same data type has different parameters, procedures serving each specific data type are designed to increase flexibility, scalability, and maintainability.

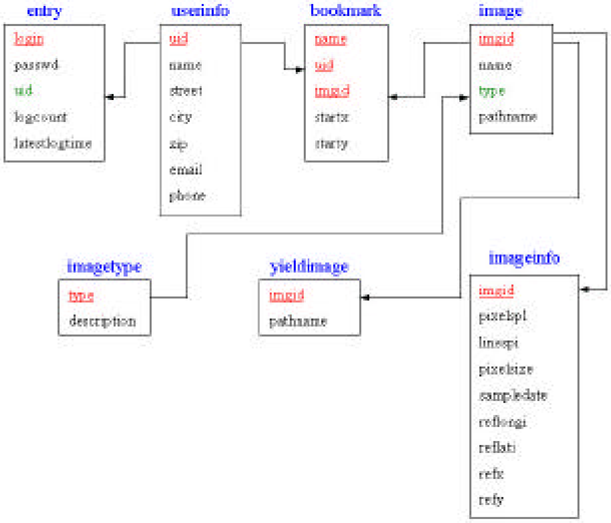


Figure 6: Database Schema

When a SMILEY data server receives a request from the SMILEY server, it will analyze the request and retrieve the header file of the required data set. From the header file, the data server can retrieve the necessary information needed to handle this data request. After analyzing the header file, the SMILEY data server then retrieves the imagery data needed to fulfil the request (either the full image or snapshot), and then formats it into what the SMILEY client requires. The resulting image is then transferred to the SMILEY client that requests the data.

Typical SMILEY Server Processing Scenario Summary

When the SMILEY server receives a data request from a client, the server will parse that request. If the request is a "Get" request for a specific image data set, then the request is dispatched directly to the appropriate SMILEY data server. Otherwise, the request is dispatched to the SMILEY DBMS process. If the request is a "Post" request, the SMILEY DBMS will update themeta-data as per instructions in the request.

When the SMILEY DBMS process is activated, it will first look for the requested data in its memory cache. If the data is in the cache, then the processes transfers it back to the client. Otherwise, it will parse the request, retrieve the indices needed to find the data from the database, and transfer them back.

When a SMILEY data server receives a request, it will first parse it. If the request is for a full-screen request (which is used to display a overview of the whole image set), it too will first look in its memory cache. If the data already exists in memory, then it returns the request immediately. Otherwise, the data server will reformat a data scene according to the user's screen size specification and transfer it back to the client after doing some additional formatting. If the request is to get a snapshot from a satellite image or yield map, the data server will just retrieve the detailed data.

SMILEY Client Processes

Snapshot Selection & Creation

Many imagery data sets are very large in size. For example, a typical TM image scene covers approximately 110 miles by 110 miles of area and consists of approximately 6200 by 6200 pixels with a resolution of 28.5 meters per pixel. This size is too big for individual processing needs; most users will only care about individual farms or towns, which would require only a small porition of the full scene. Based on the position of a sliding window, which the user moves, SMILEY will make a snapshot of the image scene before transferring it back to the client. Not only does this process help save on network bandwidth and reduce response time, but the processing requirements of the client are also greatly reduced. A snapshot can be selected in two ways: selecting directly either from the full-scene map or getting one from a bookmark that was recorded before. A screen dump of this process is shown in [Figure 7].

A typical snapshot is 250 by 250 pixels in size; using TM data, the snapshot covers 50 square kilometers. All the data mining and visualization functions described further are performed on the snapshot retrieved, not on the full scene. The main requests that a client sends to the SMILEY server are, in fact, snapshot data requests. When the user selects a snapshot area, the client sends the data request to theserver which will then process the request, passing it along to the data server. After retrieving the snapshot data, the SMILEY server forwards the snapshot back to the client.

Transfer (image filter) Functions

Transfer functions are a specialized set of image filtering/visualization functions. The SMILEY system predefines several often-used image filters: linear filter, band filter, hybrid filter, etc. SMILEY also provides a user-defined filter interface, so that the user can visually define his own filter. By applying filter functions to the image snapshot, the user can perform interactive data mining on this image and visualize the results.

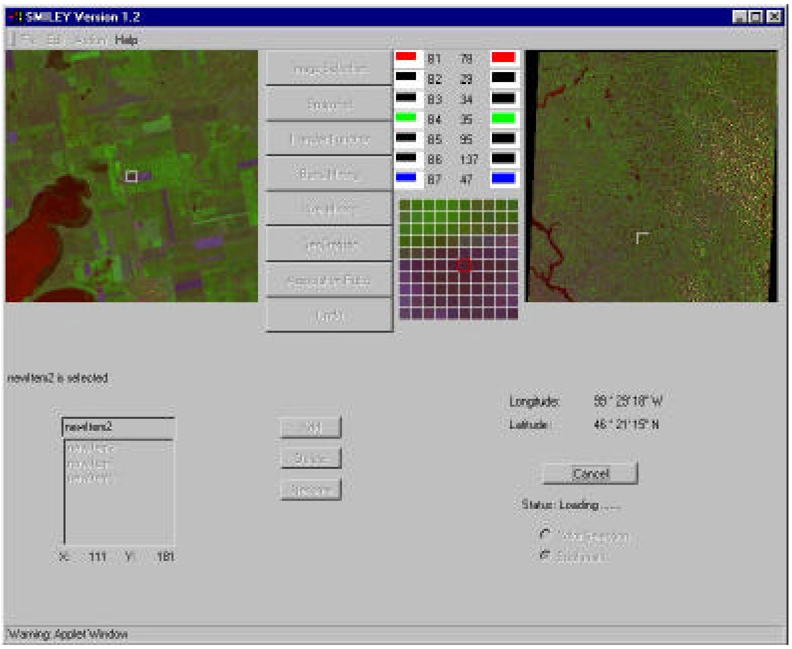


Figure 7: Snapshot selection and creation screen

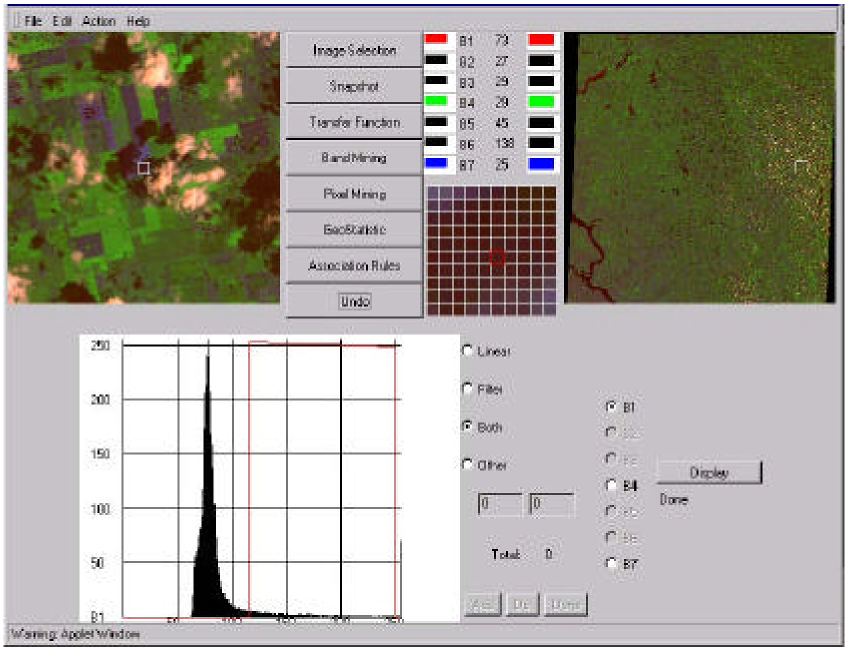
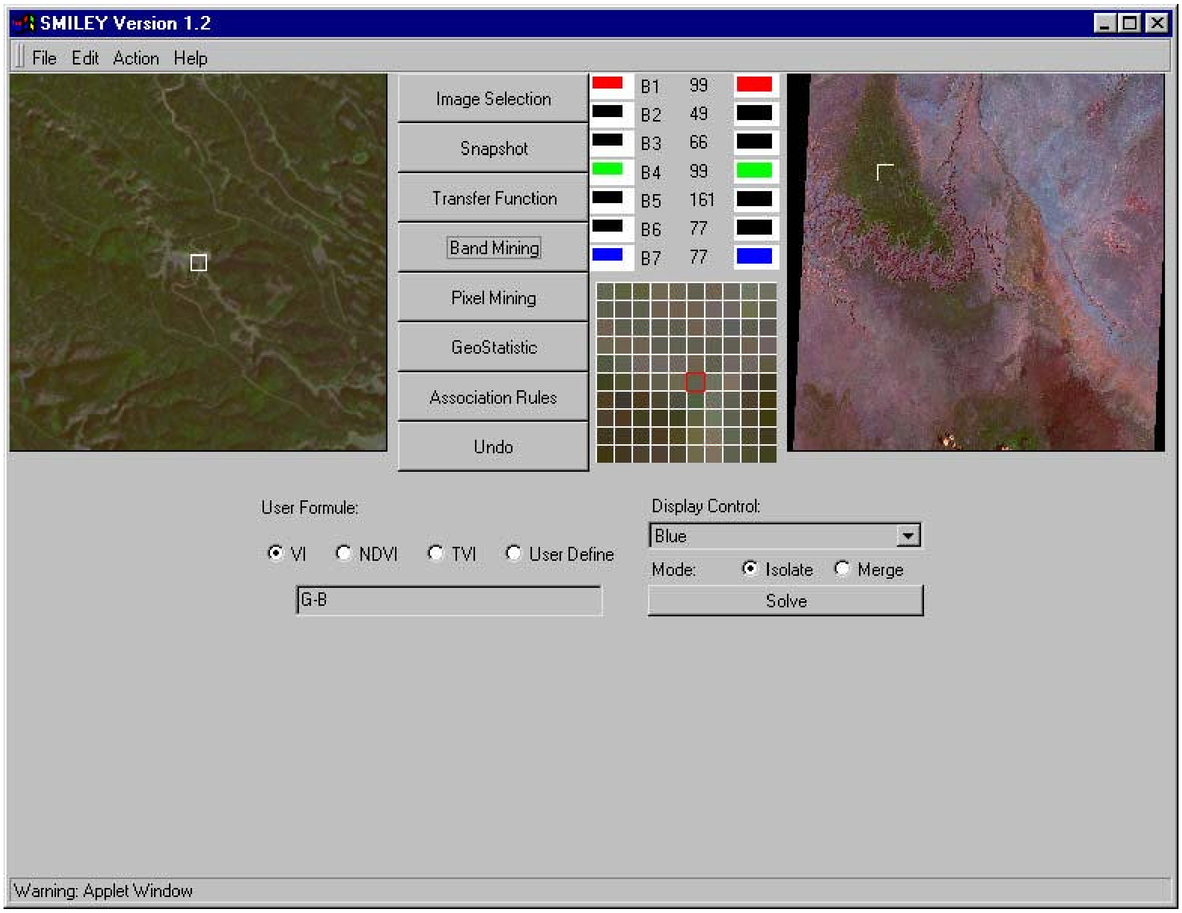


Figure 8: An sample screen of the transfer function interface

The user can also use transfer functions to find data signatures before applying band-oriented or pixel-oriented data mining functions and visually see those results.

Band Mining Functions

Band mining functions are used to do data mining based on band data. As discussed in [Background], bands of the TM image are selected to maximize their capabilities for detecting and monitoring different types of Earth resources. The user can use predefined formulas to explore a variety of information contained in the image snapshot. These functions also provide an interface for the user to enter his own formulas to generate a virtual band and visualize it. As shown in [Figure 9], the VI formula (G-B) uses TM band 4 as G (Green) and band 7 as B (Blue) to represent the vegetation index in the region. After calculating the virtual band, SMILEY can visualize the results.



🟥	B1	99	🟥
⬛	B2	49	⬛
⬛	B3	66	⬛
🟩	B4	99	🟩
⬛	B5	161	⬛
⬛	B6	77	⬛
🟦	B7	77	🟦

User Formule:

● VI ○ NDVI ○ TVI ○ User Define

G-B

Display Control: Blue

Mode: ● Isolate ○ Merge

Warning: Applet Window

Figure 9: An example screen of band mining functions

Pixel Mining Functions

Pixel mining functions are used to do data mining to detect some specific digital signatures within the image. These functions provide the major support of the data mining capabilities of SMILEY. The signatures allow the user to detect specific characteristics from the image snapshot which could represent some phenomenon in the land being sensed. To define a signature, the user selects a specific value, as well as a tolerance, for each band constituting the image. After the user specifies the signature, SMILEY will find the pixels that match the signature, and visualize the pixels after performing some additional operators (such as merge or isolate). SMILEY will also calculate various statistics of the mining operations, such as percentage of pixels within the image snapshot that matched the signature.

Geo-statistical Estimation Functions

Any type of remote sensed measurement has some resolution errors due to the hardware’ s physical limitations and the large area usually under consideration. Some statistical procedures [Knighton and Wagenet 1987] can help achieve an improved and more efficient measurement analysis. The *Kriging* *algorithm* is one type of statistical method used to predict unknown points based on the known measurements using regionalized variable theory. It is basically an interpolation process using moving-weighted averages. With kriging, the resolution of spatial data can be enhanced. SMILEY Version 1.2 implemented the semi-variance function.

Scalability and Reliability Consideration

Currently, SMILEY (Ver. 1.2) is implemented at North Dakota State University (NDSU) on a Windows-NT server. All server components (the web server, SMILEY server, and SMILEY database server), the data store, and the proc store are implemented on the same system. When client requests increase in number, or the data store increases in size, the performance of the server is expected to degrade. To solve this performance limitation, a distributed server architecture will be used to handle the increasing workload. When handling a large WWW load, the processing will be distributed over a number of low-end systems, interconnected by a high-speed dedicated network, as shown in [Figure 10]. This architecture will also provide a higher degree of reliability than a single server architecture could provide by using redundant servers.

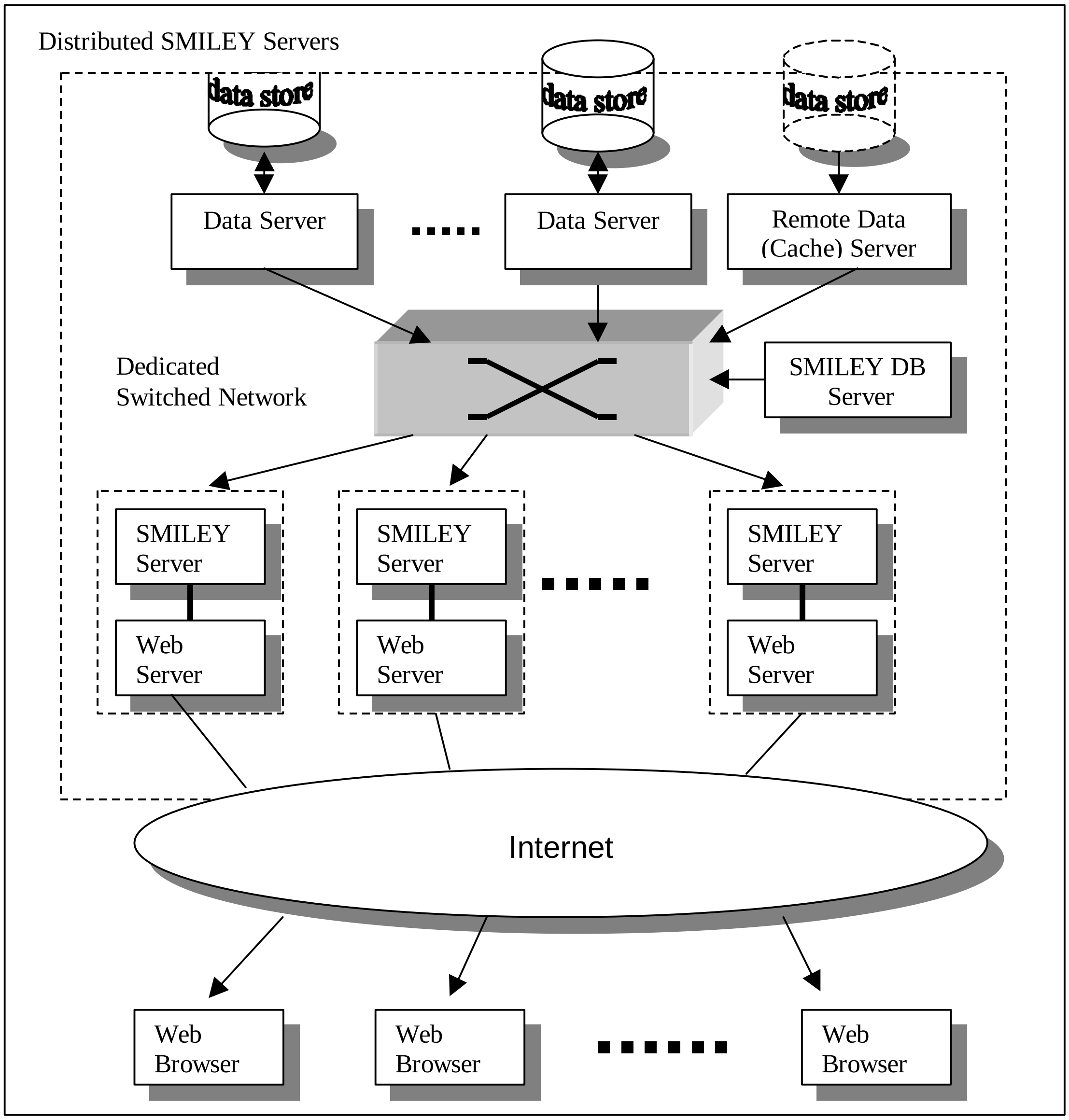


Figure 10: Distributed architecture of SMILEY

CONCLUSION

In this paper, we present the system architecture of SMILEY, a WWW-based remote sensing imagery data mining system that is currently available on the Internet (`http://smiley.cs.ndsu.nodak.edu`). SMILEY uses current Internet techniques such as Java and the World Wide Web to explore a novel method in accessing, distributing and manipulating massive remote sensing imagery data. It achieves platform independence and easy of use. The client/server model is used in SMILEY to achieve quick response times.

The current implementation of SMILEY is version 1.2. In this version, the implementation of the user interface is a Java applet, which can only be used on the Internet. The data sets to be analyzed are all stored online in the SMILEY server. The storage capacity of the server and the variety of the data are limited by the server that we are using. The geo-reference system in the current SMILEY system is also pre-mature, which will limit the usage of SMILEY for the time being.

As third party GIS systems evolve, the functionality provided by them will also increase. To utilize the power of these systems and to avoid re-inventing the wheel, SMILEY will need to provide a two-way interfaces with other commonly used GIS systems, such as ARC INFO/VIEW, MAP INFO, or IDRISI. These GIS systems have already provided a very good geo-reference system which can be used by SMILEY. What GIS systems lack is a rich collection of data mining tools, which is exactly what SMILEY provides. With the interface, SMILEY can take advantage of the functionality of existing GIS systems. Users will be benefited with this type of interaction between SMILEY and other common GIS systems.

Limited by Java applet’s security features, the current SMILEY system cannot access any data stored locally on a client system (i.e., the data that is stored in a user’s hard disk or CD-ROM). This may limit the usage of the SMILEY system to only images that are centrally indexed and accessible from the SMILEY server. Two techniques are under review to solve this problem: a standalone SMILEY system and a plug-in add-on to SMILEY. Both techniques will solve the local disk-accessing problem; but both will have pros and cons. The stand-alone technique will lose all advantages of the applet version, such as easy maintenance, upgrading, and centralized control. The plug-in technique will keep the all applet advantages but will bring in multiple binary executable code problems. When considering what the main motivation was for developing the current SMILEY system (to explore a novel architecture to access, distribute and manipulating remote sensing imagery data over Internet), the plug-in technique seems likely to prevail.

REFERENCES

[Barkstrom 91] B. Barkstrom, “A preliminary EOSDIS user model”, NASALangley Research Center, 1991

[Brach 74] E. J. Brach, “Measurement of agriculture crops by remote spectral techniques”, Remote Sensing Earth Resources, Vol. 3, 1974

[Burgess and Webster 80] Burgess, T. M., and R. Webster, Optimal interpolation andisarithmic mapping of soil properties. I. The semivariogram and punctual krigging. Journal Soil Sci. 31:315-331, 1980

[David 77] David, M., Geostatistical ore reserve Estimation. Elsevier Scoen. Publ. Co., New York, 1977

[Deering et al. 75] Deering, D. W., Rouse, J. W., Haas, R. H., and Schell, J. A., Measuring “forage production” of grazing units fromLandsat MSS data. Proceedings of the 10th International Symposium on Remote Sensing of Environment, II, 1169-1178, 1975

[Eley et al. 92] F. J. Eley, R. Granger, and L. Martin, “Soil moisture: Modeling and monitoring for regional planning”, National Hydrology Research Center, Saskatoon, Saskatchewan, 1992

[Elkington et al. 94] M. Elkington, R. Meyer, and G. McConaughy, “Defining the architectural development of EOSDIS to facilitate extension to a wider data information system”, Hughes Applied Information System, EOSDIS Core System Project Tech. Paper 194-00131. Apr. 1994

[EOS 93] EOS Reference Handbook, National Aeronautics and Space Administration, Earth Science Support Office, Washington, DC, Mar. 1993

[Huete 88] Huete, A. R., A soil-adjusted vegetation index. Remote Sensing and the Environment 25, 53-70, 1988

[Jackson 83] Jackson, R. D., Spectral indices in n-space. Remote Sensing and the Environment 13, 409-421, 1983

[Journel and Huijbregets 78] Journel, A. G. and Ch. J. Huijbregets, Mining Geostatistics, Academic Press, New York, 1978

[Juell 97] P. Juell, Implementation Issues of NDSU WWW server, Proceeding of the Fifth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford, Cal., Jun. 1996

[Krige 66] Krige, D. G., Two-dimensional weighted moving average trend surface for ore evaluation. Journal S. Afr. Inst. Min. Metall. 66:13-38, 1966

[Knighton and Wagenet 87] R. E. Knighton and R. J. Wagenet, Geo-statistical Estimation of Spatial Structure, Continuum Water Resources Institute, Vol.1 Jul., 1987

[Ronald 97] Ronald J. E., Idrisi User’ s Guide, Version 3.0, Clark Labs for Cartographic Technology and Geographic Analysis, June 1997

[Rouse et al. 73] Rouse, J. W. Jr., Haas, R. H., Schell, J. A. and Deering, D. W., Monitoring vegetation systems in the Great Plains with ERTS. Earth Resources Technology Satellite-1 Symposium, Goddard Space Flight Center, Washington D.C., 309-317, 1973

[EROS 96] EROS data center, “Thematic MapperLandsat Data”, available on Internet at URL : “http://edcwww.cr.usgs.gov/glis/hyper/guide/landsat_tm”

[Vetter et al. 95] R. Vetter, et al, “Accessing Earth System Science Data and Applications Through High- Bandwidth Networks”, IEEE Journal on Selected Areas in Communications, Vol. 13, No. 5, June 1995

APPLICATION OF ACTIVE NETWORK
IN DISTRIBUTED SYSTEMS

Faruq Rahman
Computer Science and Operation Research
North Dakota State University
Fargo, North Dakota, USA
frahman@prairie.NoDak.edu

Robert C. Gammill
Computer Science and Operation Research
North Dakota State University
Fargo, North Dakota, USA
gammill@plains.NoDak.edu

Abstract

The emerging concept of active network can be applied in a distributed system environment to expedite operations. An active network is where routers/switches perform computation on packets, not just routing or switching. Efficiency is gained by reducing the amount of packets/messages/data that need to be transmitted between end systems, which helps reduce latency and improve response-time. This paper describes a Two-Phase Commitment (TPC) operation implementation and a Condor collector enhancement using an active network simulation on an ATM networking test-bed. The Simple Network Management Protocol (SNMP) is used as the underlying data transfer protocol. A new Management Information Base (MIB) object holds the meta-data.

APPLICATION OF ACTIVE NETWORK
IN DISTRIBUTED SYSTEMS

Faruq Rahman
Computer Science and Operation Research
North Dakota State University
Fargo, North Dakota, USA
frahman@prairie.NoDak.edu

Robert C. Gammill
Computer Science and Operation Research
North Dakota State University
Fargo, North Dakota, USA
gammill@plains.NoDak.edu

1.0 Organization of the Paper

This paper is organized in sections starting with description of active network theory, its origin, and some proposed architecture. Then it discusses some possible benefits of active network, as well as bottlenecks that hinder active network research. After introduction of active networking, the paper presents two examples of distributed applications improved by active networking solution. Later, the paper provides general information on implementation of active networking and describes an underlying network message passing protocol. At the end, future work is suggested.

2.0 Active Networks

Active networking refers to the addition of user controllable computing capabilities to the data networks. With active networking, the network is no longer viewed as a passive mover of bits, but rather as a more general computation engine: information injected into the network may be modified, stored, or redirected as it is being transported [14].

2.1 History of Active Networks

The concept of active networks emerged from discussions within the board of Defense Advanced Research Projects Agency (DARPA) and the network research community during 1994 and 1995 [14]. Their discussion was on the future directions of networking systems. Several problems with today's networks were identified:

- the difficulty of integrating new technologies and standards into the shared network infrastructure,
- poor performance due to redundant operations at several protocol layers, and
- difficulty accommodating new services in the existing architectural model.

Several strategies, collectively referred to as active networking, emerged to address these issues.

2.2 Some Proposed Architectures of Active Networks

The research community proposes different methods of implementation of active networks. These methods have their own benefits and proper place of use.

2.2.1 Programmable Switches

Users would first inject their custom processing routines into the required switches. Then they would send their packets through such "programmable" nodes. When a packet arrives at a node its header (protocol type) is examined, and the appropriate program is dispatched to operate on its contents.

2.2.2 Capsules

In a more extreme view of active networks - every message is a program. Every message, or capsule, that passes between nodes contains a program fragment (of at-least one instruction) which may include embedded data. When a capsule arrives at an active node, its contents are evaluated, in much the same way that a PostScript printer interprets the contents of each file sent to it.

2.3 Possible Benefits of Active Networks

In an active network, the switches and routers of the network perform customized computations on the messages flowing through them. The idea is to capture the idle processing power of the currently high performing switches and routers, and utilize these idle CPU cycles to perform simple computations. This will help reduce bandwidth requirements for applications and thus significantly improve response-time, and network performance.

2.4 Bottlenecks in Active Network Research/Implementation

Active networking makes new demands on limited programmability available in conventional routers and switches. It requires routers and switches to perform as powerful workstations with programming environment for simple application development facilities in them. Routers and switches that are commercially available today do not provide the environment needed for users to develop and run applications on them. As a result, in our simulation of active networks, we are coupling a Sun Ultra 1 workstation with a Cisco LightStream 1010 ATM Switch. The workstation will do the computations for the switch as if it were built into it.

Some other important inherent issues arise with active network research as well. They are security, interoperability, migration strategy, etc. Due to its primitive stage of development, we are primarily exploring possible benefits at this time. Monumental enhancement of the protocol is still required for its complete implementation.

3.0 Distributed Systems

In this paper we are proposing the following benefits that an active network can offer to a Distributed System:

- reduced latency (improved response time)
- reduce network traffic (less bandwidth requirement)

Two applications are discussed in the following sections with suggested improvement using active network methods.

3.1 Two Phase Commitment (2PC)

Two Phase Commitment, in a distributed database environment, involves a master node and multiple end systems participating in a transaction. The master can commit the transaction only if all systems vote yes, else it must abort the transaction. This scenario shows the need for the master to request a vote from all the end systems and all the systems getting back to the master. For a wide-area distributed database environment, 2PC can be a very time consuming and costly process.

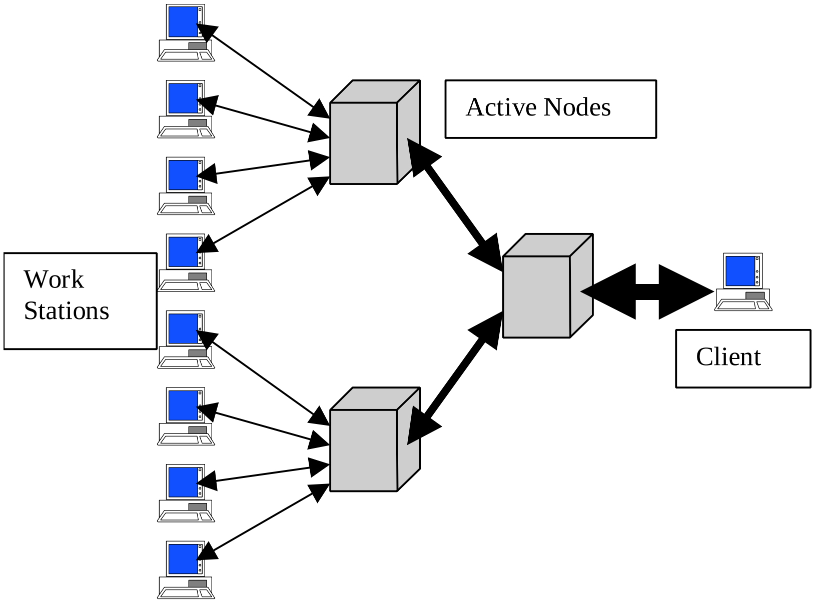


Figure 1: 2PC, traditional approach. Thickness of connection lines indicates higher traffic/bandwidth requirement.

3.1.1 An Active Network Solution for Two Phase Commitment

Here we propose a solution to the 2PC problem using an active network strategy, where active switches/routers can request and collect votes from their nearest nodes and send a combined vote to the master database system. This eliminates the need for full path communication between all nodes and the master, thus saving bandwidth and speeding up the 2PC process. This solution is based on an original idea proposed by Zhili Zhang, a graduate student at NDSU.

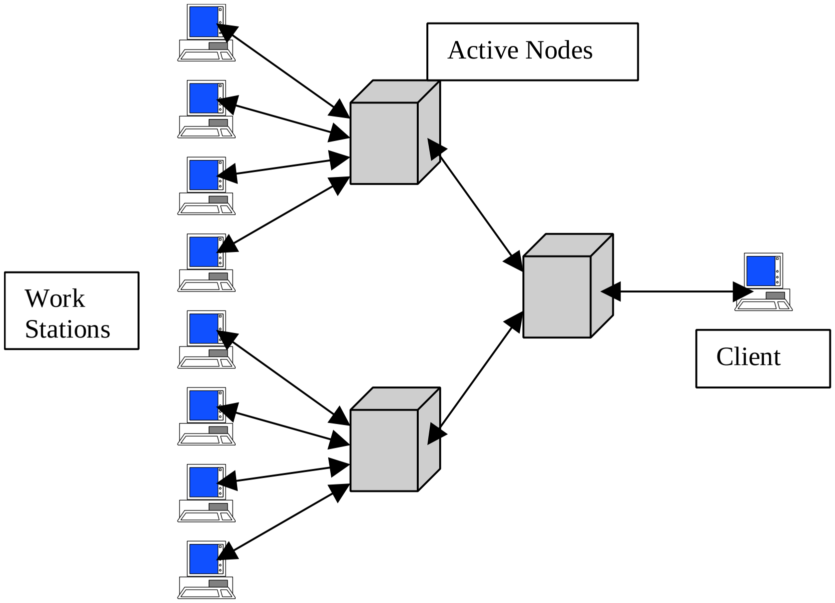


Figure 2: 2PC, active network approach. Thickness of connection lines indicates lower traffic/bandwidth requirement.

3.2 Condor - A distributed batch processing system

Condor is a software system that runs on a cluster of workstations to harness wasted CPU cycles. A Condor pool consists of any number of machines, of possibly different architectures and operating systems that are connected by a network [12]. Condor is developed at University of Wisconsin (www.cs.wisc.edu)**.**

In a Condor system two daemons run on every machine in the pool, the "startd" and the "schedd". The schedd keeps track of all the jobs that have been submitted on a given machine. The startd monitors information about its own machine, that is used to decide if it is available to run a Condor job, such as keyboard and mouse activity, and the load on the CPU. Since Condor only uses idle machines to compute jobs, the startd also notices when a user returns to a machine that is currently running and removes the job.

One machine, the "central manager" (CM) keeps track of all the resources and jobs in the pool. All of the schedds and startds of the entire pool report their information to a daemon running on the CM called the "collector". The collector maintains a global view, and can be queried for information about the status of the pool. Another daemon on the CM, the "negotiator", periodically takes information from the collector to find idle machines and match them with waiting jobs. This process is called a "negotiation cycle" and usually happens every five minutes.

3.2.1 An Active Network Enhancement of the Condor "Collector"

We have an active network solution for the Condor "collector". Traditionally, the collector has to request for a current status from each machine in the Condor pool on a given interval. When there are hundreds of such machines grouped into different clusters to be contacted for such information, this action by the collector can stress the network requiring request-reply pairs and double the bandwidth requirement. It is also a very time consuming process. By the time collector finishes collecting information on all participants and ready to make a choice, the situation can change for some machines due to the long interval.

Active network nodes can be conveniently located in each of the clusters. Each machine of the cluster can forward its current status to the active node. This node propagates the status info of its cluster towards the collector. This way the collector is getting timely information on availability of its participant machines, instead of having to pole each and every one of them. Figure 1 and Figure 2 also apply for this case and illustrate the benefits that can be achieved using the active network mechanism.

4.0 Network Message Transfer Protocol

For active network implementations we need a message transfer protocol on the network with some read/write mechanism. This protocol defines the backbone of the active network software.

4.1 Desired Criteria for the Transfer Protocol

Following are the desired criteria for the transfer protocol to be used:

- the impact of adding the underlying network transfer protocol to the switch/router must be minimal [8],
- it should not interfere with the switch/router's primary function,
- it should support multi-platform managed notes and multi-vendor switches/routers – interoperability,
- it should have simple, easy to use, standard interfaces,
- it should provide mechanisms for holding structured information,
- it should provide read/write operations for both management and managed nodes,
- it should support security

4.2 Our Choice of Protocol for the Active Network Simulation

There are a few network management protocols that offer some of the desired components. Those are Simple Network Management Protocol (SNMP), Remote network MONitoring (RMON), and Common Management Information Service (CMIP) [9]. SNMP has the most matching qualities using the UDP protocol, lightweight by nature, and supported on many heterogeneous platforms. SNMP thus became our first choice to be the underlying message passing protocol for our active network simulation.

4.3 Simple Network Management Protocol (SNMP)

SNMP is an application-layer protocol designed to facilitate the exchange of management information between network devices [8]. SNMP is used to access management information data such as packets per second and network error rates, which help network administrators manage network performance and diagnose network problems. SNMP collects information for a database called the MIB. SNMP is reliable, popular, and available on multiple platforms. SNMP is an Internet protocol, and part of larger architecture called the Internet Network Management Framework (NMF). This architecture is based on the interaction of many entities.

In the management framework, a "remote debugging" paradigm is used. Each managed node is viewed as having several "variables". By reading the value of these variables, the managed node is monitored. By changing the value of these variables, the managed node can be controlled. Management protocol operations are:

- read (e.g. snmpget, snmpgetnext)
- write (e.g. snmpset)
- a traversal operation, which allows a management station to determine which variables a managed node supports (e.g. snmpwalk, snmpbulkwalk), and
- a trap operation, which allows a managed node to report an extraordinary event to a management station (e.g. snmptrap, snmptrapd).

4.3.1 Network Elements (Managed Nodes)

Network elements comprise computers (hosts), routers, bridges, switches, servers, and other devices that are connected to networks. The SNMP Agent (or SNMP Management process) manages each Node or Element. Agents are software modules that reside in network elements. SNMP agents collect and store management information such as the number of error packets received by a network element. Agents describe the network element's state, history, and the effect of its operation [4]. It maintains a separate local database of objects for each network element.

4.3.2 Network Management Stations (Management Nodes)

Network Management Stations are general-purpose computers, and execute management applications that monitor and control network elements (or nodes) through SNMP agents. The network manager inspects the status of the network and takes action from this station.

4.3.3 Management Information Base (MIB)

Each node maintains one or more variables that describe its state. In the SNMP literature, these variables are called *managed objects*. The collection of all possible *managed objects* in a network is given in a hierarchical tree-like data structure called the Management Information Base (MIB), as shown in figure 1 [4].

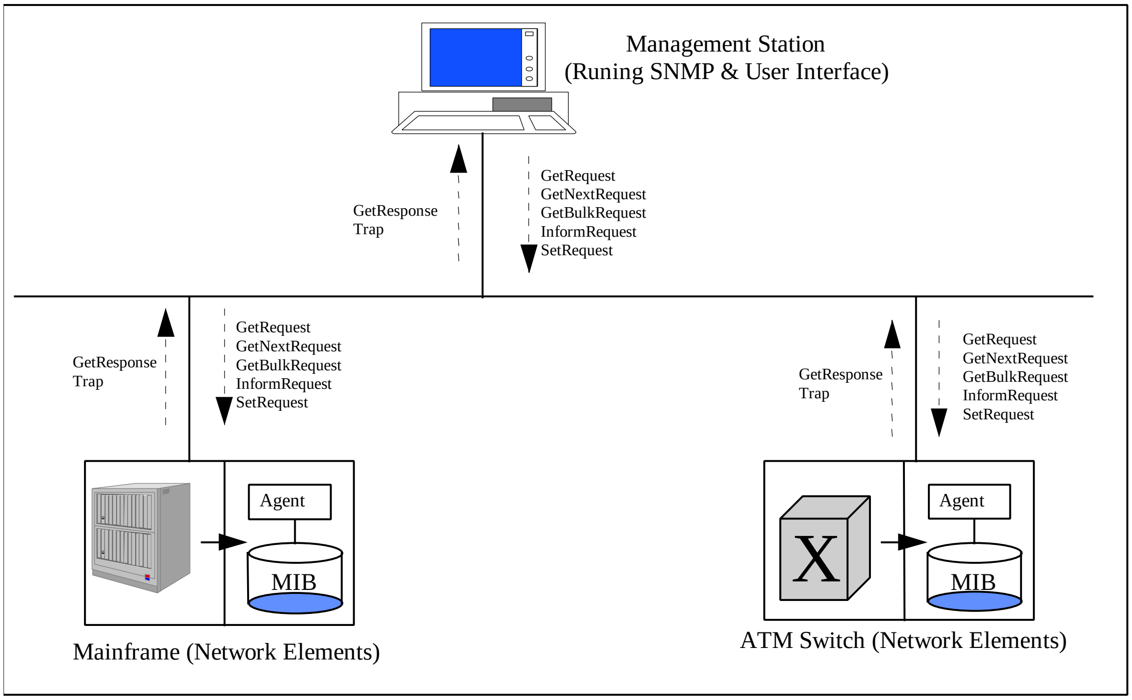


Figure 3. Management Station / Node relationships and SNMP messages

4.3.4 Example MIB (for 2PC)

ACTIVENET-MIB DEFINITIONS ::= BEGIN
-- Title: MIB module for Active Network Simulation

```
-- Date:  2/15/98
-- By:    Faruq Rahman <frahman@prairie.nodak.edu>

IMPORTS
  MODULE-IDENTITY, OBJECT-TYPE, enterprises
    FROM SNMPv2-SMI
  DisplayString
    FROM SNMPv2-TC;

activenet MODULE-IDENTITY
  LAST-UPDATED "9802241200Z"
  ORGANIZATION "Computer Science, NDSU"
  CONTACT-INFO
    "Faruq Rahman
     Postal: Computer Science Department
             258 IACC, NDSU
             Fargo, ND 58105
             U.S.A.
     E-mail: frahman@prairie.nodak.edu"

  DESCRIPTION
    "Experimental MIB module for active network research developed at Computer Science
Department, NDSU."
  ::= { enterprises 1156 }

activesimul OBJECT IDENTIFIER ::= { activenet 2 }
-- the main groups of objects in the activesimul

activesimul2PC OBJECT-IDENTITY
  STATUS current
  DESCRIPTION
    "Active Network Simulation Two Phase Commitment (2PC)"
  ::= { activesimul 1 }

-- the 2PC objects

activesimulNetLastUpdate OBJECT-TYPE
  SYNTAX     TimeStamp
  MAX-ACCESS read-only
  STATUS     current
  DESCRIPTION
    "The last time this value was updated."
  ::= { activesimul2PC 1 }

activesimulNetVote OBJECT-TYPE
  SYNTAX     INTEGER { no(0), yes(1) }
  MAX-ACCESS read-only
  STATUS     current
  DESCRIPTION
    "Indicates vote yes with value 1 and no with value 0."
  ::= { activesimul2PC 2}

END
```

4.3.5 SNMP Protocol

This management protocol is used to convey management information between agents and management stations. This protocol, defined in RFC 1157 (SNMP v1) and RFC 1448 (SNMP v2), exchanges network information through messages (technically known as protocol data units or PDU's). SNMP v2 was developed in order to increase SNMP's security capabilities and to support highly centralized network management. SNMP messages are typically carried via the UDP/IP protocol stack [11]. There are seven general types of PDU's (messages) that SNMP employs to monitor a network. SNMP version 1 defined five PDU types for SNMP: GetRequest, GetNextRequest, SetRequest, GetResponse, and Trap. SNMP version 2 adds the GetBulkRequest and InformRequest PDUs.

5.0 Two Phase Commitment Simulation

In our experiment, we have used "Scotty" [15], also known as Tnm extension, to develop the prototype of the 2PC active network solution. Tnm is an extension to the scripting language Tcl with variety of network management functions.

5.1 Brief Description of 2PC Implementation

The implementation of 2PC involves writing a MIB (section 4.3.4); writing an agent that reads in the MIB and updates it with appropriate vote for each managed node (workstation); writing additional functions for the management node (active node) that can hold votes from an array of managed nodes. Components of the agent for the management node has additional responsibility to send a request and update its MIB with values from all the managed nodes connected to it.

When the client node requests for votes to the management node, it receives a single answer from the management node saying "yes" (ready to commit) or "no" (not ready to commit). So, the management (active) node has to do simple computation to compare the votes that it stored in its MIB structure and check for an existence of a "no" vote. If a "no" vote is found, then the management node stores a resulting vote as "no" in its result_vote variable, otherwise stores "yes".

For test purpose, we have kept a vote.txt file in each workstation's /tmp directory with a vote value of 1 (yes) or (no). All the workstations have an agent running that reads the vote.txt file and update its **`activesimulNetVote`** variable with a "yes" or "no" value and **`activesimulNetLastUpdate`** variable with time-ticks value representing the last vote update time. Active node gets this status information from all the workstations it is connected to and stores it in its MIB array. Comparison is done by the active node to determine the resulting vote value that is delivered to the requesting client agent.

5.2 Note on the Simulation

Since we weren't able to develop the software on a switch or a router, our development was done on a Sun Ultra 1 workstation as previously stated. We ran the active agent on one of the Sun workstations, pretending it to be the active node, and ran collecting agents on 4 others. This simulation demonstrates that, it is promising to be able to benefit from the active networking concept. In near future, hopefully vendors will be interested enough in the new active networking concept and its benefits to allow us to develop the software on an actual switch or router by providing us with access to appropriate application development tools.

6.0 Future Work

Our 2PC active networking experiment has been very successful. It shows the potential/ need for more intensive research and development work. With support from the network vendors it can be a fruitful area of development and implementation. We are continuing our work on exploring and experimentation of active network agents to provide better solutions for distributed applications by improving their performance. We are also considering developing agents using an applet-oriented language (e.g. Java, Tcl) to provide a standard interface for future development work.

7.0 Conclusion

Despite the lack of active network implementations today, we foresee the huge potential that active network possesses. Our focus is to show some example of active network that can truly revolutionize network performance for a common type of application over distributed environment.

8.0 Bibliography

1. Welch, Brent B. Practical Programming in Tcl and TK. New Jersey: Prentice Hall PTR, 1995.
2. Vetter, Ronald J. " Asynchronous Transfer Mode": Chapter for Advances in Computers, 1997.
3. "Simple Network Management Protocol (SNMP)." http://www.cisco.com/warp/public/535/3.html (10 Oct. 1996).
4. Tanenbaum, Andrew S. Computer Networks. 3rd ed. New Jersey: Prentice Hall PTR, 1996.
5. Vetter, Roland J. " ATM Concepts, Architectures, and Protocols." Communications of the ACM v38 n2 (Feb. 1995): 30-38.
6. "Tcl/Tk Cookbook" http://www.cis.rl.ac.uk/struct/AISD/VSG/publications/cookbook/." (Feb 28, 1996).
7. "Tnm." http://wwwsnmp.cs.utwente.nl/~schoenw/scotty/man/Tnm.html (September. 25, 1996).
8. Rose, Marshall T. The Simple Book - An Introduction to Internet Management. 2nd ed. New Jersey: Prentice Hall PTR, 1994.
9. Stallings, William SNMP, SNMPv2, and CMIP - The Practical Guide to Network Management Standards. 1st ed. Reading, Massachusetts: Addision-Wesley Publishing Company, 1993.
10. Johnston, Jeffrey W. An X Window Client for TCP/IP Network Management with the Simple Network Management Protocol (MS Thesis). Fargo, North Dakota: North Dakota State University, 1992.
11. Minoli, Daniel and Golway, Thomas. Planning and Managing ATM Networks. 2nd ed. Greenwich, CT: Manning Publications Co., 1997.
12. "Condor Project Homepage" http://www.cs.wisc.edu/condor/index.html (9 Dec. 1997).
13. Pirani, Amyn and Gammill, Robert "Managing ATM Networks With A Graphical User Interface" :Small College Computing Symposium 1997 (April 18-19, 1997).
14. Tennenhouse, David L. and Wetherall, David J. "Towards an Active Network Architecture" Telemedia, Network and Systems Group, MIT (1986).
15. "Scotty - Tcl Extensions for Network Management Applications" http://wwwsnmp.cs.utwente.nl/~schoenw/scotty/ (24 Feb. 1998).

Surviving SPSS on the Desktop

Catherine E. Roraff
Initiative in Curricular Software and Support (ICSS), McIntyre Library
University of Wisconsin - Eau Claire, Eau Claire, WI 54701 USA
roraff@uwec.edu

In the last two years, the University of Wisconsin - Eau Claire has moved most of the use of the statistical software package SPSS from a mainframe to the desktop. This switch was initiated by our users and has caused changes in how UWEC provides support for SPSS and its users. The move has proved to be very challenging. This paper presents an overview of the changes that have occurred at UWEC. Issues discussed include our user base, user expectations, operational differences, documentation delivery, faculty and staff training and problems still needing solutions. By sharing some of our experiences, we hope other universities may have an easier time making the transition from mainframe to desktop SPSS.

Surviving SPSS on the Desktop

Catherine E. Roraff
Initiative in Curricular Software Support (ICSS), McIntyre Library
University of Wisconsin - Eau Claire, Eau Claire, WI 54701 USA
roraff@uwec.edu

Change is constant in the realm of technology and that brings with it the challenge of trying to adapt quickly, effectively and in the best interests of our users. The University of Wisconsin - Eau Claire has provided the statistical software package SPSS to its computer users for almost 20 years. As the academic mainframe changed from a Burroughs 5500 to two different Honeywell systems to a DEC VAX VMS system, so did the versions of SPSS. Currently UWEC's academic mainframe is a DEC Alpha Server (named, tiny) running OpenVMS and version 6.1 of SPSS.

As the University's users embraced desktop computing, software companies like SPSS adapted their products to this environment. Early versions of SPSS did not appear to offer our users enough incentives to make the switch from the mainframe product to the desktop product. With the introduction of SPSS/PC+, approximately version 5, the University's users began to take notice and ask for access to the software. Novell servers provided initial access for most faculty, staff and student labs. When SPSS version 6.1 was made available, most users switched from the DEC Alpha server and declared their preference for the SPSS desktop product.

Supporting SPSS on the desktop has proved to be as challenging as coping with the pace of change in desktop technology itself. Unlike the mainframe environment where only one version and one platform needs support, the desktop environment is not homogeneous and will probably never be homogeneous. Budget limitations mean there will always be old and new hardware. Curricular and discipline specific requirements mean there will always be multiple platforms. Different hardware and different platforms results in the of use of and need to support multiple versions of SPSS. The version of SPSS used depends primarily upon the hardware configuration (memory, hard disk space and network server connection) and the operating system. UWEC is currently running SPSS versions 6.1.3 (Windows 3.11), 7.5.2 (Windows 95 and Windows NT) and 6.1 (PowerMac).

Operational and delivery system changes were needed to support this new environment of multiple platforms and multiple versions. The most significant operational change for users was the switch to the use of SPSS portable files instead of SPSS system files. An SPSS system file limits its users to the version of SPSS on which the system file was created. If the version of SPSS differs between a faculty member's office and the classroom or lab in which SPSS will be taught or demonstrated, the system file may not be usable. Students often find that their favorite lab is not available for use because an instructor has a class scheduled and they must move to another lab which may have a different version of SPSS. By persuading users to save all their data files as SPSS portable files, problems of file access from different versions is minimized. Overall, our users readily saw this advantage and accepted it quickly. Using the portable file format also simplifies putting datasets out for access on Novell and NT servers. Only one file has to be created and maintained. So far, the only drawback of these portable files is their slightly larger size.

Teaching the desktop version of SPSS to new users is much easier and takes much less time than the old FORTRAN based command language version of SPSS. The graphical user interface makes the desktop version parallel the look of other products with which most users have had experience. Workshops for new users of the DEC VAX version of SPSS are a minimum of four hours and workshops for new users of the desktop version of SPSS are about an hour to an hour and a half. Faculty have found that they can incorporate an SPSS exercise (with a predefined dataset) into their course in only one class session. Users generally find the pull-down menus and menu screens of the desktop version extremely similar to applications like word processors and spreadsheets. Therefore, the users easily make the transition to the new version since they can relate using the product to an application they already know and use, even if it is only Windows 3.11 itself.

Datasets must be handled differently if they are very large such as the cumulative General Social Survey or if they are on 9-track magnetic tape. Large datasets are more effectively manipulated and maintained on the DEC Alpha server. Regular dat tape backups are done of the Alpha and other servers. Disk space is readily available on the Alpha in comparison to other servers. Set-up and processing of the large datasets, especially if they come with command files for SPSS, is efficient and does not tie up a desktop machine for long periods of time. Access to 9-track magnetic tape is limited to UWEC's administrative Unisys mainframe. Staff from the administrative support unit will unload files from a tape so that they can be ftp'ed to either the DEC Alpha server or to a desktop machine. FTP is an essential utility for moving files around in our multi-platform, multi-version environment. Teaching how to use ftp may need to be done depending upon the assignment or the research project.

FTP combined with web access and Internet searching allows users to more readily find and retrieve datasets. Many governmental units and research groups such as the Bureau of Labor Statistics and ICPSR, which used to provide datasets for study via 9-track tape, CD-ROM or diskette are now making these available from their web sites. The downloading of datasets may be restricted via a passworded account but the index of available materials is usually searchable by all users. Identifying and locating datasets via the web is an important step forward in getting more students especially undergraduates involved in doing research using SPSS.

FTP also allows UWEC to readily move portable datasets to Novell and NT servers for access from almost any computer on campus. Mapping to a network drive is a relatively new concept for many of our users but it can be taught easily and has not appeared to be a problem for most users. UWEC's users are encouraged to use floppy disks for storing important documents, mail and other files. Therefore, they are comfortable with the concept of changing their default drive to a. With SPSS datasets, instead of changing to drive a to open their document or mail, they change to drive j which has the datasets stored in folders organized by departments, then instructors.

Documentation delivery also changes with desktop SPSS. No longer is the big maroon manual supreme or the status symbol to be carried all over campus! Users don't want expensive manuals which change with every version and which are rarely at their fingertips when they run into a problem. Users want immediate answers to their questions by using online documentation which is incorporated into the software package. UWEC has been loading the syntax reference guide (approximately 20 meg) with our standard installs. With the built-in help and the syntax reference guide, most users are able to find the answers to their questions without ever exiting SPSS. The addition of the Statistics Coach help section has met with mixed reviews at UWEC. The purchase of manuals has declined steeply but more users appear willing to purchase a text about using SPSS in their specific discipline. The SPSS web site has an extensive list of books and manuals grouped according to discipline. Instructors at UWEC have been acquiring evaluation copies of some of these titles. At least two titles that were evaluated this way are now available for purchase in the University bookstore. One instructor has a small, inexpensive paperback text as a required purchase by his students. The SPSS web site also has a series of white papers which discuss specific topics in depth. Some of these white papers can be used to supplement the standard documentation.

SPSS also maintains an electronic help desk where users can e-mail questions. UWEC does not promote this for student use but faculty and staff have made use of this to get additional help. As SPSS site coordinator, I try to act as a buffer between UWEC's users and SPSS Inc. including their phone support line and e-mail list. This strategy appears to be advantageous to both the users who get faster and more site specific assistance and SPSS who has one point of contact for UWEC. In most cases when I contact the SPSS support staff, I have already tried everything I can think of to solve the problem and have all the information they need ready to be e-mailed or faxed so they can replicate the problem or question.

There are still operational and delivery system changes that UWEC have not been adequately addressed. Many users when they are initially exploring a data set, want to perform a “frequencies variables=all” run. These users are used to having the results of this run printed out. Large volume print jobs on desktop systems are a problem for UWEC. Since most of our public access and networked departmental printers are laser printers, these large volume jobs are much more costly than when they were run out on a mainframe printer. Toner cartridges and paper expenses quickly accumulate when users have multiple 100+ page printouts. These printouts also tie up a printer for long periods of time inconveniencing other users of that print station. If I can’t dissuade users from doing these type of runs or from looking at the results on the screen and printing out only those tables of interest, I try to get the users to do these runs on the DEC Alpha server. If a user is not familiar with using SPSS on the Alpha, I will set it up and do the run for them. It is a relatively simple process. A user will e-mail me the file or send it to me on diskette via campus mail. I convert it to a portable file and ftp it to the Alpha. I import it into SPSS and run the job and send the output back to the user via campus mail or they can pick it up at the test scoring desk at the administrative computing center. Long-term UWEC will be trying to set up one of the administrative mainframe printers as a printer than can be chosen by users with large volume SPSS print jobs.

Color printing is also a problem for UWEC. There are very few color printers on campus. Some of the color printers are restricted access for faculty and staff only and others are on a cost per page basis. With the impending release of version 8 of SPSS, UWEC expects more requests for color printing of the graphs and charts that can be produced with SPSS. Release 8 is primarily a re-write of the graphics modules of SPSS. Supposedly not only is the quality of the graphs improved, there are also more graphs and charts to choose from (e.g. 3 dimensional). Currently we try to get users to change fill patterns and colors so that when charts or graphs are printed on a gray scale printer, the different chart elements are distinguishable. This is not well accepted by the user community and will probably become even less accepted as the new release gets installed. Users currently expect to include graphs and charts in the presentation of their data. No long term solution has been agreed upon as yet.

The most dramatic change to accompany the switch to SPSS use on the desktop was the change in the number of users and in how SPSS was being used. SPSS has been, and still is, essential to much faculty research and to certain discipline specific methodology courses. The redefinition of the baccalaureate degree which was implemented approximately two years ago has placed greater focus and emphasis on undergraduate research and on capstone courses and that has resulted in a much broader SPSS user base. One of the tools promoted for undergraduate research is SPSS because it is readily available to the students, it is relatively easy to learn and is appropriate for a variety of research projects. Capstone courses also promote research using statistical analysis and graphical presentation of data as a way to show the integration of acquired skills in their disciplines. UWEC is also promoting more faculty, staff and student collaborative research. The overwhelming majority of the students involved in this research are undergraduates. Collaborative research with colleagues from other campuses is also growing. Network access, especially the use of electronic mail and file transfer (ftp), has simplified the mechanics of collaboration and is an area of great potential growth.

Distance education also presents another challenge. Currently only University owned equipment have the SPSS software installed on it. As more and more students attend classes without being physically on campus, access and authentication problems must be addressed. For example, UWEC has a graduate program in nursing and most of its graduate students are not living in Eau Claire. Many use out of area Internet providers to access campus resources which means UWEC cannot just limit access by its range of assigned IP numbers. How can they access SPSS to perform the analysis for their theses? Authentication for remote users encompasses not only access to software like SPSS but also access to some University library resources. It is a priority problem for the University.

Not only has there been a large jump in the number of users of SPSS, there has been an increasing number of first time users. Students in 100 and 200 level (freshman and sophomore) courses are given assignments to construct a hypothesis and test that using a dataset which the instructor provides. In some cases, these types of assignments present problems because an instructor has assumed that the student has already had a basic statistics course. (Many students at UWEC postpone that course until they absolutely have to take it.) In higher level courses, students are being asked to construct a survey instrument, collect data, define and enter the data and analyze and present their data. Presentations may be done in class or as part of UWEC's Research Day. (Research Day allows our undergraduate students to gain experience making conference setting poster sessions to a panel of judges drawn from other institutions and corporations without the expense of leaving campus to attend a conference.)

There is also a trend emerging that the research being done by the undergraduate students is not just frequencies, crosstabs and chi-squares. Although this seems to be the starting point for most student projects, many have proceeded on to perform ANOVA and regression analysis and to learn how to present their information graphically. In some instances to do the graphic presentation, students export SPSS data into a geographical information system (GIS) or other graphics packages.

Both the increase in sheer number of users and the pattern of use of more sophisticated statistical testing has presented a support problem for UWEC. Demand for support has increased dramatically and many of those requests are on advanced statistical procedures. I am the only support person assigned specifically to the statistical packages and there are only two applied statisticians on staff (both faculty in the Mathematics department). Many departments have staff who teach research methodology for their disciplines but many of these staff have had little or very limited experience or training with the new version of SPSS. Last year with funds from a UW system grant, UWEC was able to offer a series of workshops for twenty faculty that introduced them to the basics and some of the advanced capabilities of the new version of SPSS and to resources on the web which could be used with SPSS. Providing professional development to faculty and staff on each new release of SPSS continues to be a problem for UWEC.

Training for the campus SPSS site coordinator may also be an issue. One potential solution to the training of a site coordinator like myself is the subscription training offering of SPSS. I have used and supported SPSS since I came to UWEC in 1979 but from January 1990 until April 1996, I was not directly involved in supporting SPSS. I took on the position of primary SPSS support and the role of SPSS site coordinator upon the resignation of another staff member. At that time, SPSS for Windows was just gaining favor with the users and I had no experience with it. As I started to work with SPSS for Windows, I realized that there was much that I needed to learn, including a crash refresher course in statistics. In March 1997 I wrote a professional development grant that was funded by our Office of University Research. This grant paid for the SPSS subscription service fee - $1200. My department agreed to pay for my hotel and meals and I agreed to pay for my transportation. The subscription service allowed me to enroll in as many courses as I wanted to during a twelve month period (4/1/97 - 3/31/98). As expected, these class expenses strained my departmental budget but I was fortunate that retraining for me on SPSS was considered important not only to the department but to the University as a whole.

I was able to attend a total of nine courses presented by SPSS, all but one at SPSS Inc. in Chicago. These classes not only allowed me to become familiar with the current version of SPSS for Windows but it also allowed me to review statistics and survey research techniques. I was also able to bring back to campus the SPSS course guides which were used to teach these classes. (These guides are now available for purchase as an alternative to attending the course.) These guides have become invaluable reference materials for me. Not only do they have examples and datasets which are not included with the SPSS distribution media, they also provided me with an outline of how to teach everything from basic topics in SPSS for Windows to advanced statistical techniques. I was also able to meet other individuals who perform the role of SPSS site coordinator and to meet with the University's account representative at SPSS.

Surviving SPSS on the desktop requires a new attitude which includes even more patience than you think you can possibility possess, flexibility and a willingness to change and adapt constantly. It is a very different world from mainframe computing where support staff and computer center staff are in control of the system. In the desktop environment, our users have much more control over their desktops and appear to take over the “ownership” of their desktop (unless they have problems, then it becomes “ours” again). Our users are more vocal and more demanding. They are more likely to experiment and work on their own or with friends and colleagues. Demands may include wanting a new version of SPSS before it has even arrived on campus. Experimenting with SPSS procedures they haven’t used before or don’t really understand may result in some unusual e-mail messages or phone calls asking for explanations of the output. Exporting SPSS data or output into other software packages like MS Office, a GIS or advanced graphics software is becoming much more common. User expectations of the support staff seem to have no limits. I can no longer tell a user, you can’t do that and have them accept that. If I say, you can’t do that in SPSS, I am usually met with the response: well, what software can do that for me and, by the way, I need it for class on Friday. Some things never change!

Acknowledgments

Financial support for the SPSS classes was provided under an Academic Staff Professional Development Grant, Office of University Research, University of Wisconsin - Eau Claire.

"A Team Approach to Multimedia Design, Development, Testing and Evaluation: NDSU's new Center for Academic Information Technology"

Dr. James B. Ross
Associate Director, Information Technology Services (Learning Technologies)
Interim Director, Center for Academic Information Technology
North Dakota State University
United States of America
ross@plains.nodak.edu

Abstract

The Center for Academic Information Technology (CAIT) is promoting a process that brings students, staff and faculty together as productive teams. CAIT projects, because they are done outside the classroom, foster new relationships between students and faculty by undertaking collaborative projects outside of the classroom. The students work in groups, headed by full-time staff members, who investigate the emerging technologies and help the students deploy them. Our goal is to help faculty, staff and students change the way they interact with instructional technology on NDSU's campus. Student employees are a key to achieving this goal. The CAIT has automated many of the tasks that were once assigned to student employees. The CAIT will still construct web pages for faculty, for example, but we will not assign a student to work long hours on a professor's course pages if all the professor wants is the student's labor. The students that the CAIT employs are too gifted for us to squander their talents on work that faculty could arrange to have done in other ways. It is, frankly, too expensive to employ students to work below their abilities.

This paper focuses on some of the new management techniques employed by the CAIT during its first year of operation and demonstrates some of the latest technology that our staff has designed to support the university's mission.

"A Team Approach to Multimedia Design, Development, Testing and Evaluation: NDSU's new Center for Academic Information Technology"

Dr. James B. Ross
Associate Director, Information Technology Services (Learning Technologies)
Interim Director, Center for Academic Information Technology
North Dakota State University
United States of America
ross@plains.nodak.edu

The New Center

The NDSU Center for Academic Information Technology (CAIT) is the one-stop shop for instructional design and multimedia design, development, testing and evaluation. It opened in summer 1997 with two new full-time staff: an Instructional Designer and a Multimedia Technologist. The CAIT supports around fifty student, staff and faculty projects at any given time during the year. The CAIT accepts projects in the following areas:

Basic Training	Course Design	Content Enhancement	Licensing
Customized Training	Animation Development	WWW Site Design	Graphic Design
Computer Programming	Instructional Assessment	WWW Site Management	Marketing and Distribution
Technical Support	Instructional Evaluation	Project Management	Copyright Release

Full Time Staff

The CAIT Interim Director is also the Associate Director of Learning Technologies in Information Technology Services. The Interim Director is the initial contact for multimedia proposals from students, staff and faculty. The Interim Director, the Instructional Designer, the Multimedia Coordinator, the NDSU Webmaster and the Multimedia Technologist assess proposals, assign priority and resources to the projects and create a timetable for project design, development, testing, and evaluation.

Instructional Designer

The Instructional Designer helps faculty translate course objectives into instructional modules, identifies appropriate learning strategies and learning technologies to enhance teaching and learning, and designs and implements formative and summative evaluations of instructional technology.

Multimedia Technologist

The Multimedia Technologist designs web-based, database-driven courseware and stand-alone multimedia, graphic interfaces, coordinates project team tasks, and trains student employees.

CAIT's Student Employees

The CAIT promotes a process that brings students, staff and faculty together as productive teams. The projects are done outside the classroom, to foster new relationships between students and faculty. We intend to produce a different classroom environment using the work that students and faculty undertake collaboratively outside of the classroom under the direction of the Instructional Designer and the Multimedia Technologist.

The students work in groups, headed by full-time staff members. The CAIT staff investigates the emerging technologies, but the students help faculty deploy them. Our goal is to help faculty, staff and students change the way they interact by insisting the faculty collaborate with students in the production of instructional technology. Student employees are a key to achieving this goal.

First Model of Collaborative Teams

The design, development, testing and evaluation model needs to be flexible in order to maximize student and permanent staff effort. All CAIT projects are executed as team projects in order to take advantage of the diverse talents which the student bring to CAIT. The CAIT is also responsible for keeping abreast of emerging technology and incorporating the best practices in learning technologies in the life and work of the campus. Consequently, CAIT student employees, as part of the team assignments, will investigate national and international trends in adopting, developing and deploying learning technologies. Team building is very difficult and has been as big a challenge to the staff as producing the multimedia itself. We are developing our third model in less than one year. At first we organized teams to take on parts of each project. We planned to manage he most involved projects using all of the following teams:

Intake Team, comprised of student employee(s) and Instructional Designer, to help promote the project by sketching objectives, determining necessary skills and resources, and proposing specific design and development strategies (the proposer will then make formal application to CAIT's Interim Director for support).

Project Selection Team, comprised of student employee(s), Instructional Designer and Multimedia Center Coordinator, which trains the proposer (as necessary) and appoints team leaders.

Design Team, comprised of the Instructional Designer or Multimedia Center Coordinator or Multimedia Technologist and student employee(s), which defines the instructional strategies, chooses delivery technology, locates media and sketches the user interface.

Development Team to do the graphics and interface design and computer programming.

Testing Team and Evaluation Teams comprised of Multimedia Center Coordinator or Multimedia Technologist and student employee(s) to assure that the development satisfies defined criteria and objectives (includes assessment of the development process as well as assuring that the product is deployed successfully in the classroom or at a distance).

We tried assigning student employees with one set of skills (an animator, for example) to the promotion team on one project while asking the same student to evaluate another project, and at the same time complete work as development project leader of a third. Teams were supposed to form and disband as projects began and ended. We knew that some projects would be so simple and straightforward that they skip from promotion to development. A Web page is a good example because it required only an initial meeting of the client with the Instructional Designer and the Multimedia Technologist and one sitting with a student programmer and the Multimedia Center Coordinator to complete the project. Others will take more time and planning. Simply stated, clients' projects which required more resources would take more time and necessitate more in-depth project management. The difficulty came in breaking the mold which shapes most student and faculty interaction. Faculty would rather assign a whole project to one student, let the student frustrate through to a solution and submit a product for review. The students are used to this model and like it because it allows them to focus all of their attention on the computer screen. Both groups are reticent cooperators or collaborators.

"Over the past 6 months, the CAIT has struggled to find its place," according to the Instructional Designer. "It was conceptualized to be a process-oriented group, with projects leaders heading each phase of development, and coordinating the efforts of the project team. While this model works in theory, it has not been easy to implement in practice. Projects do not always occur in discrete phases, at least when dealing with instructors and course development. It is often a hurry up and wait situation."

"Increasingly [the Instructional Designer has found herself having] to manage all phases of a project, including interface design and production. It may be that [our early] model is not appropriate for development of course development." Several recent changes have helped turn teams around.

Breaking the Mold

New Clientele

We no longer assign a student to work long hours on a professor's course pages if all the professor wants is the student's labor. The staff and students have constructed image libraries and course web page templates. Faculty, then, build customized course content from databases containing WWW components, including a QuizGenerator, On-line Discussion Forum, Dynamic HTML WWW page builder, navigational button builder. The students that the CAIT employs are too gifted for us to squander their talents on work that faculty could arrange to have done in other ways. It is, frankly, too expensive to employ students to work below their abilities.

The CAIT has decided to focus much of its human and material resources on a few faculty, staff and student projects. We accept proposals for highly interactive, content-rich WWW sites or multimedia. The most ambitious program along these lines is called Academic Technology Partnerships (ATP). Faculty who join ATP develop high content/high interactivity course materials using the latest information technologies. ATP faculty have the entire CAIT at their disposal each summer. Some of the ATP's work is featured in the presentation of another paper in these PROCEEDINGS, "Year Three of NDSU's Instructional Web Project," and URLs are attached to the "accomplishments" section near the end of that paper. The Vice President of Academic Affairs selected eight faculty to participate in ATP this past summer and asked the CAIT to expand the program for the coming summer.

A New Work Environment

For those faculty who insist on assigning work to students rather than working along side them as co-producers, we have built a physically separate, walk-in multimedia center where work can be left for later pickup. Some students, too, are still more comfortable working with faculty using the standard classroom model, so we assign them to the walk-in center.

The CAIT has also moved into a larger work space that is designed to facilitate collaborative teamwork. The new room accommodates more student employees at any one time, so the supervisors can assign common work schedules to team members. The new room houses the offices of the students' supervisors, so staff are more intimately involved in the students' day-to-day work and the staff are more accessible to the students. The room also is configured so that staff, students and clients can leave the computers and discuss storyboards or layout plans. This last change sounds trivial, but it is perhaps the most significant. Much of the work we see getting done in the room is accomplished simply by turning toward another team member and away from the computer. The room is now laid out to accommodate this type of activity where previously the staff had to encourage it.

Current Model of Collaborative Teams

For the time being, students are assigned to either Dynamic HTML, stand-alone multimedia, WWW page, UNIX programming or WWW Course design and development teams. Each executes work assigned and prioritized by the Instructional Designer and Multimedia Technologist. Each team has scheduled training, often conducted by the students, which serves to keep individual students abreast of trends and cognoscente of timelines and priorities. Some students serve as project and team leaders, as assigned by full-time CAIT staff. This new model has produced some very involved Java, Java Script, Perl, and CGI programming and WWW-based tools for electronic course development. The student teams use the following application development tools on a rudimentary basis: Claris Filemaker Pro 4.0, Adobe Exchange, Acrobat, Illustrator, PageMaker, and Photoshop; 3D Studio Max, Macromedia Freehand, Extreme 3D, Director and Flash.

Examples of Accomplishments to Date

Representative examples of CAIT projects are listed below and discussed in the presentation of this paper. A complete list may be found at:
http://webmaster.cc.ndsu.nodak.edu/reviews/cait/index.shtml

Project Description	Project Director	Client Department
JAVA-based dose-response module	Elizabeth Smith, Instructional Designer	Pharmacy / ATP faculty
Virtual Model United Nations	Elizabeth Smith, Instructional Designer	Political Science / ATP faculty
Course WWW generator -- allows faculty to dynamically generate their web pages on-line	Ludvik Herrera, NDSU Webmaster	CAIT / faculty
Transfer Equivalencies Database, developed to provide information on the transfer course equivalents between NDSU and other universities in the United States.	Felix Guerrero, Multimedia Technologist	Office of the NDSU Registrar
Course Index Database, allows professors to add, delete and replace links to their WWW instructional material.	Felix Guerrero, Multimedia Technologist	CAIT / faculty
On-line QuizGenerator -- Allows professors to create online quizzes. Students take the quiz via the web and are able to see a graded quiz after submitting the quiz. Scores are stored in a database for later retrieval.	Felix Guerrero, Multimedia Technologist	CAIT / faculty

The direct outcomes of employing collaborative teams in the CAIT include:

1. Electronic course materials with rich content and a high degree of interactivity between the student and instructor, among students and between the student and the course materials that support improved student learning and demonstrate the role of NDSU as a leader in electronic course delivery;
2. A faculty with skills to improve course materials using the latest information technology tools that can be adapted to other aspects of the university mission;
3. Assessment techniques to determine the effectiveness of information technology upon improving student learning;
4. student employees tell us that they have learned how to get things done. The "how" involved:
   a) collaborating with co-workers
   b) getting help from others who worked in other groups
   c) constantly acquiring and applying new knowledge
   d) working under supervision and learning to supervise

The Next Team Innovation

We will continue to work on the team concept. According to the Instructional Designer, "We need several [additional] discrete groups. For example, 1) a stable group devoted solely to investigating and developing emerging technologies, 2) one group to meet demands of getting syllabi and other supplemental (i.e., quizzes) materials onto the web, and 3) a smaller group with more clearly defined roles & skills to work with faculty on a larger scale for course development (i.e., like having major portion of their course on the web)."

The Instructional Designer plans on establishing an "electronic media documentation, testing, evaluation and training" group within the CAIT. The group will help her assure faculty that the tools and the course materials that the CAIT helps them use are effective. She has already set out some the group's objectives: "evaluating instructions for tools, navigation of web pages and tools, download time on and off campus across both MAC and PC platforms, and general beta testing of all products prior to their release."

Preparing Computer Science Graduates for the PC World

John P. Russo
Department of Mathematics and Computer Science
Indiana University South Bend
South Bend, IN 46634
jrusso@iusb.edu

Abstract

When computer science students are hired, employers will typically expect them to be knowledgeable, if not experts, in microcomputer environments. However, students who major in computer science often spend most of their laboratory time in a UNIX environment. Consequently, many CS students graduate with little or no experience working with PCs. The paper discusses possible solutions to this dilemma and focuses on the creation of a special course, "Advanced PC Techniques". The course is designed to provide students with many of the tools they need to face the PC world with confidence. Various aspects of the course design are discussed, including selecting topics, finding textbooks, and choosing appropriate lab exercises. The paper discusses the author's experiences teaching the course and future directions that the course might take.

Preparing Computer Science Graduates for the PC World

John P. Russo
Department of Mathematics and Computer Science
Indiana University South Bend
South Bend, IN 46634
jrusso@iusb.edu

Introduction

The IUSB Computer Science Department, like many others around the world, is UNIX based in its laboratories and most of its courses. Although our students get some microcomputer experience in the first two courses (via working with Turbo C++), their later lab experience is usually in a UNIX environment. Given their general knowledge and probable encounters with microcomputers, most CS graduates will be reasonably comfortable as PC*users*. However, many companies will expect more than user-level abilities. When computer science graduates are hired, many employers will understandably expect them to be "power users" and to have extensive PC operating system and programming knowledge. The lack of such knowledge tends to undermine a job seeker's self-confidence and may at times lead to embarrassment. Since microcomputers are now ubiquitous in the workplace, CS departments should consider giving the PC environment the attention it deserves.

Partial Solutions to the Problem

We can encourage students to buy their own microcomputers. Given the low price of microcomputers, many students are already purchasing their own machine. I've also found that a number of students have thought about buying their own machine and respond positively to a bit of encouragement. A student who owns a PC will probably get some hardware experience and will have a good chance of becoming an expert user. As noted in the introduction, however, employers generally expect more than this. Of course, some students simply cannot afford to purchase their own microcomputer.

We should consider integrating the PC into more courses. Many CS departments already use the PC in some of their courses. For example, I have my assembly language students work on PCs. For this to be possible, of course, there must be PC labs available in your institution. At IUSB, we do not have a departmental lab devoted to PCs, so students who do not own their own computer must work in public labs. However, these labs are often crowded and students avoid them when at all possible. We have set aside some space in one of our UNIX labs for a few PCs running Windows 95. These machines are very popular and have given some students valuable PC experience. However, there is no guarantee that merely doing programming assignments on a PC will give students an adequate depth in this environment.

A Special Course for PC Mastery Provides a More Complete Solution

It is clear that for many students, the approaches above provide only a partial solution to the problem stated in the introduction. Our department decided that a special course should be created to address the problem. The goal of the course, which we called "Advanced PC Techniques," is to help students become PC experts. The remainder of this paper is devoted to describing this course and discussing my experiences.

Choosing Topics for the Course

The following are some of the topics that came to mind when I started to design the course.

1. Architecture and history of 8086 family
2. Intel assembly language
3. PC Hardware, low-level programming
4. MS-DOS, batch files
5. Windows 3.1, 95
6. Windows programming
7. Installing PC hardware peripherals
8. Networking PCs

It was clear that not all of the topics could be covered adequately in a three credit course. Topic 7 was eliminated because of laboratory limitations, although a few students elected to do a hardware installation lab project. Topic 8 was eliminated because it was so extensive and because we offered a separate networking class.

Designing the Course

Finding a conceptual framework that would provide a natural order for the diverse topics seemed hopeless at first. However, after a bit of thought, I realized that one could order the topics from the lowest level of abstraction (hardware) to the highest level of abstraction (Windows). Historical information, such as the history of the IBM PC, could be given at almost any point in the course. In the end, I covered the topics in the order in which they were listed above.

Finding a textbook proved to be more difficult. In the end no single textbook proved to be adequate. I eventually chose three books for this course.

> *Peter Norton's Inside the PC,* by Norton, Eggebrecht and Clark. This book is geared mostly to the user, and contributed to making students "power users."
>
> *The Peter Norton PC Programmer's Bible* by Norton, Aitken and Wilton. This book is programmer oriented (as was the course*.)* It contains a wealth of programming information specific to Intel based PCs.
>
> *Teach Yourself Visual Basic in 21 Days* by Gurewich, was used during the Visual Basic portions of the course.

The books were not all used in the same way. I assigned reading from *Peter Norton's Inside the PC* in the early part of the course, primarily to bring them up to speed as users. *The Peter Norton PC Programmer's Bible* was used mostly as a reference for some of their programming assignments. *Teach Yourself Visual Basic in 21 Days* is written in a self-study way and was used for Visual Basic programming assignments.

I also gave out a list of books that I had found useful and suggested they might consider adding to their collections, depending on their interests. Here is the list.

Upgrading and Repairing PCs, Que Corporation

Programmer's Guide to Low-Level Functions and Interrupts, Sams Publishing

DOS 6.2 Simplified, Boyd & Fraser Publishing Co.

Windows 3.1 Bible, Peachpit Press

Windows 95 Secrets, IDG Books

Visual Basic SuperBible, Waite Group Press

After some discussion, we decided that an appropriate prerequisite would be our course in computer architecture (which includes assembly language). In teaching the course, I found that this requirement was adequate.

Laboratory Work

The programming assignments were designed to reinforce the topics discussed in lecture and were an important part in the course. These included the bulleted items below.

- Assembly language programming (done at the DOS level).

This helped support the material on the Intel 8086 family architecture and was done using the Turbo C++ built-in assembler, BASM. This allowed students to do their editing and debugging in the Turbo IDE, with which they were already familiar. My goal was not to make them assembly language experts, but to provide them with a solid introduction to ASM-86 as a way of understanding the architecture of the Intel microprocessors. About half of these assignments involved writing standalone ASM-86 procedures that were called from Turbo C++ programs. The students also learned how to use “asm” statements within their C++ source code. This allows CPU intensive sections of a function to be written in assembler and remaining sections to be written in C++. Here is an example of a routine that does this.

```
/***************************  OneBits   ***********************************

DESCRIPTION  Finds the number of 1-bits in a 16 bit int.

PARAMETERS   N  Prototyped as unsigned, but can be any 16 bit integer data
                type.

RETURNS      The number of one bits in the binary form of N. For example,
             if N == 7, then N == 0000 0111b, so N has 3 one bits.

NOTE         The function is written mostly in ASM-86, so it relies on
             the ability of the C++ compiler to assemble such code.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
```

```
int  OneBits(unsigned N)
{
  int OneBitCount;

  asm  mov ax, 1              //  Move bit mask into ax register.
  asm  mov cx, 0              //  cx will be used to hold the number of 1-bits

CheckNextBit:

  asm  test ax, N             //  Perform AND, to see if we have a 1 bit
  asm  jz   NotOneBit         //  Bit was not a one bit.
  asm  inc  cx                //  We found a 1 bit.

 NotOneBit:
  asm  shl ax, 1              //  Adjust the bit mask.
  asm  cmp ax, 0              //  Have 1 bit moved all the way left?
  asm  jnz CheckNextBit

  asm  mov OneBitCount, cx    //  Store in local variable for return

  return OneBitCount;
}
```

- **Low-level C++ programming (done at the DOS level).**

This supported the material on PC hardware and gave students some experience working with interrupt routines. One assignment involving their using software interrupt 33h, which provides various low-level mouse routines. Another assignment involved directly accessing video memory. A third assignment taught them how Hewlett Packard PCL is used to control a printer.

- **MS-DOS batch programming.**

Most of the students had no batch file experience and knew little about MS-DOS. They did an assignment that involved writing several simple MS-DOS batch files. Due to changes in the PC world, I spent less time on MS-DOS the second time I taught the course. I will probably spend even less time the next time I teach the course.

- **Windows programming (and introduction to event-driven programming)**.

The first time I taught the course, students did very little Windows programming, but by the time the course was offered for the second time, the desire to get some Windows programming experience was one of the major reasons that students enrolled. Consequently, about 40% of the course was devoted to Windows and Windows programming. Because of its ease of use and its popularity, I chose Visual Basic as the programming language. Early assignments were taken directly from *Teach Yourself Visual Basic in 21 Days*. I believe that getting an introduction to event-driven programming was one of the more valuable aspects of the course.

Student Response and Course Evaluation

Last spring, about 30 students signed up for the course. Given that typical enrollment in one of our *required* upper-level classes is about 12, and that Advanced PC Techniques is not required, this level enrollment is quite impressive. By the end of the course, 25 students remained, 20 of whom were present when I handed out course evaluations.

In answer to the question “Overall, how would you rate the course?”, 15 chose the highest rating, “Excellent”, and 5 chose the second highest rating, “Good”. The written student comments were generally favorable: here is a sampling.

> “Excellent subject matter-- many items not generally covered in other classes - mouse, video, Intel ASM programming, Visual Basic, etc. VB should be offered in other classes so advanced items could be covered.”
>
> “More Visual Basic, less assembly language.”
>
> “I found all quite interesting, but was particularly fond of time spent in assembler.”
>
> “This course provided a major opportunity to acquire new skills that are beneficial for the real world.”
>
> “All around excellent course, it should be required.”
>
> “Maybe spend a bit less time on ASM. Maybe a little more on Java. Liked covering so many topics, liked all the topics-- now if I had enough time to really delve into all of them...”

Future Directions for the Course

The PC world changes more rapidly than most other computing environments. Consequently, I expect the course to change considerably each time it is offered (every two years at IUSB). Windows now totally dominates the PC world and MS-DOS will play a less important role in future offerings of the class. Although the Internet is not “owned” by the PC world, the WWW and related ideas have become the “hot topic.” The first time I taught the course (1995), I did almost nothing with WWW topics. The second time (1997), I spent time talking about html documents, Java and JavaScript. When I teach the course in 1999, these topics will no doubt play a much bigger role in the course. Finally, if I can work out the logistics, I would like to have students do more hardware based labs, such as installing disk drives or network cards.

Development of Daily Homework and Quiz Manager (D.H.Q.M.) for Enhancing Teaching/Learning in Large Classes

Nem W. Schlecht
Information Technology Services
North Dakota State University
schlecht@plains.nodak.edu

Dr. Sudhir I. Mehta
Mechanical Engineering Department
North Dakota State University
mehta@badlands.nodak.edu

Abstract

Research indicates that assigning daily homework problems, providing quick feedback, and obtaining quick assessment of student learning are important in enhancing the teaching-learning process. However, performing these activities in a large class of over a hundred students is extremely time consuming. This paper describes a tool, The Daily Homework and Quiz Manager (DHQM), that has been developed to implement the above activities fairly easily, benefiting both students and instructors.

Object-oriented Perl and certain features of Netscape, make the user screens of the DHQM package attractive, easy to understand, computationally efficient, and quickly delivered to the user. Making sure that the user receives the information quickly is an important aspect that is too often overlooked in many of the fancy web based applications today.

The use of Perl allows the DHQM package to be easily portable between different UNIX platforms with few, if any, changes necessary. The Perl code provided in the DHQM package follows Internet standards on distributing Perl code packages and object-oriented authoring and testing methods. Since methods for class data retrieval and manipulation are provided, instructors can easily extend DHQM to suit their specific needs. Perl has long been admired by system administrators and, for some time, has also formed the backbone of many CGI applications on the Web. By using Perl, which is familiar to many programmers and Web developers, we hope that the DHQM package may be extended even farther by other parties at other institutions.

A portable system, small code size, fast data delivery, extendible code, and an easy user interface are all part of the DHQM package. This paper covers a brief history and future potential of the DHQM package.

Development of Daily Homework and Quiz Manager (D.H.Q.M.) for Enhancing Teaching/Learning in Large Classes

Nem W. Schlecht
Information Technology Services
North Dakota State University
schlecht@plains.nodak.edu

Dr. Sudhir I. Mehta
Mechanical Engineering Department
North Dakota State University
mehta@badlands.nodak.edu

Introduction

In 1994, only a year after I was hired by Information Technology Services at N.D.S.U., I was asked to work with Dr. Sudhir Mehta to develop a program to help him with his classes. Through much hard work and many meetings we managed to come up with the original D.H.Q.M. system, which was in use for 3 years. The original system, although declared a success, was missing some features and was cumbersome to use. Also, after 3 years, it was getting a little behind the times. This paper describes some of the process of developing the new system, descriptions of the CGI programs used in the system, and some of the features of the new system.

The Original D.H.Q.M. System

The original D.H.Q.M. System, designed and written in the fall of 1994 was a strictly text-based system. Some readline functionality was implemented to help the students and instructors, but this was only in the form of tab-completion of certain list items (such as a list of dates for which scores were available). A very small menu system was developed for the instructor, but it was difficult to use. The system was not user-friendly by today's standards. Also, it was required that all students and instructors have accounts on the same UNIX server, which posed many problems over the last 3 years. In the end, a mainly unused server was set up for the purpose of handling the D.H.Q.M. system and to give it a centralized location.

Features of the Latest D.H.Q.M. System

The new D.H.Q.M. System retains all of the functionality and instructor tools of the old systems but is now more user friendly and diverse. Students and instructors can now access the system from any machine running any operating systems. Also, the graphical web form interface is a common and well understood interface, so no new form navigation knowledge is required. There is more persistence of data of the instructors, as well. The statistical breakdown given to the instructor when they process a new day's scores are saved and available at any point in the future to the instructor. Finally, students can now request to see any combination or all of their grades for a full semester. In the old system, they needed to request them one at a time.

The Development of the Latest D.H.Q.M. System

This past summer, with the aide of the North Dakota State University Technology Fee and a proposal by Dr. Mehta, the new D.H.Q.M. system was developed. The main goal was to create an easy to use interface for both the students and instructors. This goal, by itself, would have been easily accomplished in a short amount of time. However, I took advantage of this by reworking the system itself into a more object oriented system. In the original system, I had grouped some subroutines into perl libraries and also had a menu system developed. I used these as a framework for my object classes and the layout of the main instructor's tool page, respectively. Unfortunately, not as many object oriented functionalities were put into place I would have liked. There are cases where inheritance should be used to help simplify some of the CGI programs. Also, some sections of code in the CGI programs should be generalized and placed into classes as methods. The main system, as a whole, has progressed greatly. The complexity in making all of the scripts 'web' aware has been significantly hidden by the use of object oriented perl.

The D.H.Q.M. Perl module package

The DHQM Perl package is available for download from the main D.H.Q.M. page here at N.D.S.U. (see Appendix). It follows the same standards as other module packages that can be downloaded from any CPAN sites. It is compressed with GNU's compression program gzip and archived using the UNIX tar command. Once extracted, you must run "perl Makefile.PL" in the extracted directory and then run "make". You may also want to run "make test" after building is complete to make sure that everything is working. Finally, run "make install" to install the software. Perl standards have been employed as much as possible, but are lacking in several areas. There is no embedded documentation suitable for use by the *podtoxxx* converters. There are comments, but you have to look at the class description code to read them.

The 'DHQM::Class' Perl Module

This object class is used for most of the high level things that the instructors deal with. It has methods to list the classes that an instructor is allowed to modify, return references to associative arrays to students' names and grades, and make updates to both the data file and the control file. It also handles the rotation of the files (in case of possible instructor error or a major malfunction). Lastly, it also provides methods for interacting with the daily statistics file.

The 'DHQM::Code' Perl Module

This class deals with the alpha codes that students receive. The most important method is a recursive one that creates the list of alpha codes for a class. Any size class can be used with the D.H.Q.M. project. Code creation is done in intervals of powers of five. For example, if you have a class of 20, you only need an alpha code of length 2 because you can have 5^2=25 codes with 2 alpha digits. For classes of 125 or less, you need 3 alpha digits (5^3=125), etc. In our test classes, we have always just picked 4 alpha digits so that there are no worries about overrunning the 125 code limit with 3 alpha digits. There is no reason you can't use 4 alpha digits, even if you only have 10 students. I should note at this time that we haven't tried to use more or less than 4 alpha digits for our alpha codes, so it is a bit untested. I know that they work with the Code class, but they may not with some of the CGI programs listed below. Other methods that are provided by the Code class include converting from alpha digits to numeric digits (a=1, b=2, etc.) and a method to increment an alpha code's value (abcd => abce).

The 'DHQM::Config' Perl Module Methods

This class only contains a constructor method and one variable. This is the only module that programmers have to modify. It contains the path to the main library directory where the D.H.Q.M. site configuration and class data is all stored. The Class, Instructor, and Student classes all create an instantiation of the Config class since they all require or manipulate information from the main D.H.Q.M. library directory.

The 'DHQM::GProc' Perl Module Methods

This module does the actual calculations for grading the homework and quiz questions. For quiz grading, one must call the set_quiz_key() method and pass it the quiz key, a string of characters that is compared to each students' quiz answer. A method to convert the homework score is also located here. The homework score is based on a 10 point system. If a homework question is 100% correct, a student receives a mark on the 'a' circle. For each letter past 'a', the student receives one point less. The lowest score a student will receive on the homework is 5, unless that student did not do the homework.

The 'DHQM::Instructor' Perl Module Methods

This module only contains two methods. First is a method that retrieves the name and scores for a specific student (via alpha code and class). The other method updates the grade data portion of the class database for a specific student. The latter method does not do an update for a specific date. The calling program must figure it out using the methods provided in the Class class.

The 'DHQM::OMR' Perl Module Methods

This module has only one method, which processes the O.M.R. data line by line. The O.M.R. data is the file that is created by whatever optical mark reader hardware you may have. This module may need to be modified to work with your optical mark reader. It is impossible to know all the different output formats that O.M.R. hardware may produce. It is up to the local programmer to rewrite this module to handle their specific O.M.R. data format.

The 'DHQM::Stats' Perl Module Methods

This module has methods to calculate some basic statistics. These include: min, mean, max, total (calculate sum), freq (frequency distribution), and std_dev (standard deviation). There are statistics modules available from the C.P.A.N. sites, but they usually do higher level statistics. I only needed basic statistics for the D.H.Q.M. package.

The 'DHQM::Student' Perl Module Methods

This module has two methods. The first one is for the student grade page and does both authentication and returns all the grades for the student for the class. Verification of students is done with a combination of alpha code and their N.A.I.D. number, which is a unique identification number for the student that is given by the university. The other method calculates some statistics (mean and total) for the student's homework and quiz scores.

Performance Tweaks

The D.H.Q.M. package takes advantage of the WIDTH attribute of the IMG tag to improve the performance of loading pages. All of the graphs that appear on any of the D.H.Q.M. pages are actually just a 1 pixel by 1 pixel GIF image that has been stretched to 10 pixels using the HEIGHT attribute and some width using the WIDTH attribute. All graphs are done inside of TABLE cells and always represent a percentage of some type. Because the graphs represent a percentage, that percentage value is inserted directly into the WIDTH attribute. Therefore, the value that the graph represents is easily viewable by comparing the size of the graph to the overall size of the table cell that it is in. Unfortunately, to get this behavior, the student or instructor must run Netscape 4.0 or higher.

Student CGI Programs

The following is a list of the CGI Programs used in dealing with students. There aren't very many, since the problem of handling the student data is very easy and the students can only do one thing - get their grades. This is; however, a likely location to add additional features such as an online quiz or test.

index.cgi

This is the main page that comes up when students access the student grade area. The only reason this is a CGI program is that we need to present the students with a list of classes to which they might belong.

studgrade.cgi

This CGI program does the actual lookup of the Student scores. It presents the students with a list of dates for which there are class scores and always gives the student their cumulative homework and quiz grades as well as the class averages for these. The student may then select any or all presented dates, ask for an update, and receive their homework and quiz scores, as well as the class averages, for all the days selected.

Instructor CGI Programs

The following is a list of the CGI Programs that are tools for the instructor. They include programs to both update and extract data from the class databases, as well as programs to obtain detailed information about the class or statistical information about how the class responded to the homework and quiz questions. A much larger amount of time was spent on updating and enhancing these programs. Most of these programs, as well as the student program, studgrade, set up certain state information via HIDDEN input fields and ’recursively’ call themselves. Only if an instructor teaches more than one class will they be asked to select a class in any of the scripts below.

addinit.cgi

This program does the initialization and setup for a new class that an instructor will be teaching. At this time, a class list including the names of all the students as well as a unique identifier for them must be provided. The user codes for the student are also generated at this time and displayed for the instructor. The instructor may wish to print this out in order to give the students their codes. However, I would suggest that the instructor download the class database (available on the instructor’s tool page) and load it into Excell or some other spreadsheet where they may format the list to their liking.

addscore.cgi

This is the heart of procomr program. It does pretty much everything magical in the whole system. Using the above objects, it grades the homework and quiz scores, updates the database with those scores, performs the statistical analysis, and displays status information on the processing. This program took the most time to develop.

addstud.cgi

This CGI program is used to add students to the class list. It is just the opposite of delstud, allowing an instructor to add students that have just added their class to the class database. The new students’ grades for days that they have missed are all set to ’-1’, the same as if they had been signed up since the beginning at hadn’t addended class. If the instructor gives makup homework and quiz questions to this student, they can change the absentee grades with the use of the chstudgrad program.

chstudgrad.cgi

The chstudgrad program is used to change the grade for a student for a certain date. After the instructor selects the class, student, and date, a web form is presented with the student’s current grade already filled in. The instructor simply makes any changes and applies them.

classdb.cgi

This is the program that actually does the work for dldb. It collects the data from the proper class and does split() calls, etc. This is the program you will want to change if you want your downloaded data to be something other than tab-delimited. It should pop up a download box by default. Of course, this depends on the server’s *mime-types* file and the instructor’s *.mailcap* file or Netscape mime handler. The file is sent as the ’application/octet-stream’ mime type and will be saved as *classdb.cgi*.

delstud.cgi

This CGI program is used to remove students from the class list. As we all know, students change their curriculum often or change majors. Since unattending students will bring the class averages down, the instructor has the option of removing them from the class database.

dldb.cgi

Many times an instructor will want to load the grades into some other application. The dldb program can be used for this purpose. By default, it asks the instructor for any parameters and calls classdb (see above).

multday.cgi

This program presents the instructor with a multi-day summary for all of the students in a class as well as the totals for all of the students. Up to 5 days can be selected. Students that have missed a class or had a problem with their opscan sheet will be marked in the table with a different background to distinguish them from the other scores.

procomr.cgi

This CGI is just a wrapper for addscore. It sets up the form to be filled in by the instructor. As other programs, it is a CGI program so that it will prompt the instructor only with the classes that that instructor teaches.

studsum.cgi

This program displays the same statistical analysis graphs as the addscore program. It allows the instructor to view data on a certain day's scores, even if that class was held months ago. It does not go out and re-do any analysis. The output from the analysis that was done when the data was loaded is saved with each class' data. This program merely loads that data, fills out the table, and displays it.

D.H.Q.M. - The Next Iteration

The development of any software package is an iterative process. I have described to you only the second iteration of a software package that I have no doubt will continue to both be used and developed further. This past summer, while doing the conversions from both a text to a web interface and old-style perl library to new-style perl objects, I became aware of many shortfalls and problems with the current system. My hope is that others will help me in this endeavor and that the D.H.Q.M. package will continue to evolve. I personally wish to implement a standardized object to handle to I/O of the current system, so that a relational database or other storage management solutions can replace the current, slow, colon delimited data and control files. I also hope that the D.H.Q.M. package can be made to interface with other instructional web tools that others are developing.

Appendix

Official D.H.Q.M. Homepage: http://www.ndsu.nodak.edu/dhqm/
Official Perl Site: http://www.perl.com/
For a list of CPAN Sites: http://www.perl.com/CPAN
Official Netscape Site: http://www.netscape.com/

Biographies

Nem W. Schlecht is a UNIX Systems Administrator and Programmer for the North Dakota State University System. His specialties include Perl programming and database applications. His responsibilities include technical Web administration and systems administration for Silicon Graphics workstations. He also is a consultant for faculty, staff, and students on a wide range of UNIX related topics. He is currently a Senior in Computer Science at North Dakota State University.

Sudhir I Mehta is a professor of Mechanical Engineering at North Dakota State University. He has 3 years of industrial and 19 years of academic experience in the areas of engineering education research, instrumentation, controls, robotics, design optimization, and machine vision. He has developed 2 CDROMs containing hypermedia based instrumentation and communication resource modules. He has also developed innovative techniques for active learning and quick assessment. Dr. Mehta received the Carnot Award for the best teacher of the year from the students of Pi Tau Sigma Society four times. He also won the Carnigie Foundation's 1997 North Dakota Professor of the Year.

Web and Database Integration for the Delivery of Dynamic Information (on a Budget)

Nem W. Schlecht
Information Technology Services
North Dakota State University
schlecht@plains.nodak.edu

Abstract

The idea that every faculty & staff member should learn and write HTML for the delivery of their departmental and class information is unreasonable. Most have attended seminars, been given fancy editors and technical handouts, and had other resources at their disposal. They may attend or use these resources with high hopes, but many are still disappointed and frustrated. Many departments want to solve this problem, but as in many small colleges and universities, funding is somewhat lacking. This paper describes the structure of a system which implements a solution to the above problems, with little strain on faculty or most staff or on their departmental budget.

First, a brief, non-technical introduction will be given, with a review of some similar implementations with various delivery methods (Gopher, newsgroups, etc.) and why these methods do not address all of the current problems and needs of departments. Next, a description on the usage of free software packages and how to obtain and implement them is addressed.

This is followed by a discussion of the underlying relational database implementation and some possible database schemas to use in that database. This database is the backbone of the organizational structure that is needed to solve the above problems. Then, we look into the ways the above packages can be integrated to work together in providing an information solution that is both easy to update and to obtain current, valid information from. An example of some working models as well as an in depth analysis of these models and some of the problems with them will conclude the technical discussion.

The paper concludes with a exploration of how far this model can be taken and what options there are to expand its capabilities and functionality.

Web and Database Integration for the Delivery of Dynamic Information (on a Budget)

Nem W. Schlecht
Information Technology Services
North Dakota State University
schlecht@plains.nodak.edu

Introduction

Computer science, as a profession, deals with data. We collect it, examine it, do statistical analysis on it, give that data to someone else, etc. With the explosion of the Internet, we now have a lot more data that we are moving around. A lot of us have set up mail filters to handle the hundreds of e-mail messages we receive each *day*. We browse the Web looking for information for our research or maybe to find out what is going to be on T.V. at 3:00 in the morning. Unfortunately, a lot of us are having either a hard time keeping up with all this data, or we work around the clock trying to stay ahead of the game. Before the creation of the web, the three major information exchange forums were E-mail/Listserv, Net News and Gopher. I'm sure all of us are still using and will continue to use E-mail, as well as I'm sure most of us either are or have been signed up to at least one Listserv list. Although effective, E-mail and Listserv take up a lot of bandwidth and disk space - each user in a list gets a copy of the message. Net News was favorable because it had two major enhancements to Listserv. First is the idea of an information thread. Somebody could make a statement, someone else would add something, another person would disagree, and the first person would explain their statement further. Any person could easily read the entire *thread* of discussion and understand what all was happening. They did not have to search through a hundred mail files to find out what everybody said. Net News also had central storage, reducing both the amount of network traffic and disk space needs associated with such discussions. Lastly, Gopher, or the text-only web, which was the first application with an easy to use *fetch* interface. Instead of sending data out to everybody, they had to come to them to get it.

All three have advantages and disadvantages, but they all suffer from the lack of being dynamic. Gopher had the ability to run a program or fill out a simple form, but its capabilities were severely limited. Today, with Javascript, Java, Perl, and a host of other specialized applications, programs, and languages, information sources can be very dynamic. Instead of writing static or mainly static HTML pages, we should concentrate on the creation of page templates with database backends. Once we have our template, we can easily update the underlying information and make it instantly accessible to web visitors. Alternatively, we can update our template and in an instant we can have a new 'look and feel' to our web page. Also, searching for specific data is easily done, leaving little need to create and maintain a search engine. Dynamic web delivery has been going on for a long time now with *server side includes* allowing the current time or the last time a certain page was modified automatically being sent to the user. This paper discusses taking this to the next level and making more of your web pages dynamic.

Free Software

Contrary to what many I.T. managers may say, free software is a viable solution to many problems, and they are actually the preferred solution by many systems programmers such as myself. Also, many departments that have a tight budget will probably have to venture into the 'free' market whether they want to or not. A lot of free programs actually have excellent capabilities and support. MySQL, mentioned below, is an excellent example of this. For implementing a web and database integration solution, I feel UNIX is the best operating system for the job. It is robust, usually faster, and usually much more stable than Windows or Windows NT. Also, there are free versions of UNIX available, such as Linux or FreeBSD. For CGI scripts, Perl is an excellent choice. I prefer to write my programs in an object oriented fashion, where applicable. Perl has O.O. features as well an interfaces into countless other extensions. For web and database integration, there are 2 modules that are crucial for this. The first of these is the CGI module, which allows for easy web form creation and processing. The other is the DBI module (Database Independent Interface), which provides a cross-database Perl API. The DBI module requires an accompanying DBD (Database Driver) module. There are DBD modules for a host of commercial databases, including Oracle, Sybase, Informix, Ingres, IBM’s DB2, as well as others. The free databases that it supports include MySQL, mSQL, and PostgreSQL. Of these, I prefer MySQL, which is supported on a wide variety of UNIX platforms as well as on Windows NT. It makes use of threads for better performance compared to mSQL and PostgreSQL. PostgreSQL has more features, but is slower due to the overhead in managing those features. mSQL is the slowest with complex queries and has the least number of features, but is nice and lightweight. Because mSQL is lightweight, it may be a good solution on slower hardware where memory is precious. For a web server, I use Apache, which is used by roughly half of the web servers on the Internet. It has a rich feature set, but is not so bloated as to cause system performance degradations. Also, Apache has the ability to load modules into it to extend its functionality. For example, there is a Perl module which allows the web server itself to execute Perl code without having to fork off an additional process.

One thing to keep in mind is that with Perl and the DBI module and DBD modules, pretty much any web server and database can be used. If you want to start out using MySQL and then later migrate to Oracle, you shouldn’t have too much trouble in either moving your tables and data or in updating your Perl scripts. The same thing is true if you start out using Apache and decide later to start using Netscape’s Enterprise web server. Also, all of the above mentioned free databases are implemented in a client/server manner. Therefore, if you want to run your web server on one machine and have your relational database on another machine, you shouldn’t have any problems.

There are other solutions that I have seen implemented using software other than Perl and using the CGI & DBI modules. Another favorite is PHP/FI, which is a pseudo-Perl language. It is mSQL and MySQL aware and the PHP/FI code is inserted directly into your HTML files and parsed with either a CGI script or an Apache module. As always, programming in C or C++ is an option. Most databases have a C client library and API that Perl or PHP/FI take advantage of for their database connections.

Relational Databases

Why use a relational database? Why not a ’flat text’ file or some other type of file? First is speed. Even mSQL, the slowest of the free databases, will enhance your searches dramatically. Next is the concern of file locking. Many web applications today are completely ignorant of the fact that there may be multiple copies of themselves running. Without proper file locking, files get corrupted, data is lost, or programs will deadlock and crash or never exit. Relational databases are usually programmed from the bottom up to handle multiple requests at the same time properly. Another problem of flat files and other files is the fact that searching them or changing them is often a very tedious and long process. Many of us have run across a tab or colon delimited file and found that manipulating a certain ’column’ of data required a lot of extra code to do splitting and joining of the data. With a relational database, column searching and updating are easy. Finally, although it is sometimes a pain, normalizing your data to be inserted into a relational database makes you re-examine your goals and your data. By going over what data you are working with and what you want to do with it, you will probably gain an intimate understanding of the data and be able to make additional tweaks and optimizations that you would have otherwise overlooked.

Designing a relational database schema is pretty easy if you’re not too concerned about ’normalization’. When you normalize your data, you remove repetitive, large data samples and replace them with unique keys. For example, in the schema example below [Fig. 1], there are no users listed in the *cr_data* table. Instead, each user is assigned a unique numerical identifier in the *users* table, and that id is stored in the *cr_data* table instead. We have now changed a 20 to 30 byte name into a single byte. For small tables, this doesn’t make much sense since it adds an additional burden onto you to always have to do a table join. However, on large tables, you can save megabytes of disk space rather easily by normalizing your data in such a manner. Searches will also be quicker on any field in the table since less information will have to be copied from disk to memory to examine it.

Database Schema Example - System Change Report

This is an example of the normalized database schema used in the "System Change Report", a web application that I’ve developed for tracking changes to machines within my organization. I’m not sure if it’s normalized to the ’5th level’ or not, but it’s pretty complex. In doing lookups for a certain change report, 10 queries are done! However, those 10 queries take only fractions of a second to complete.

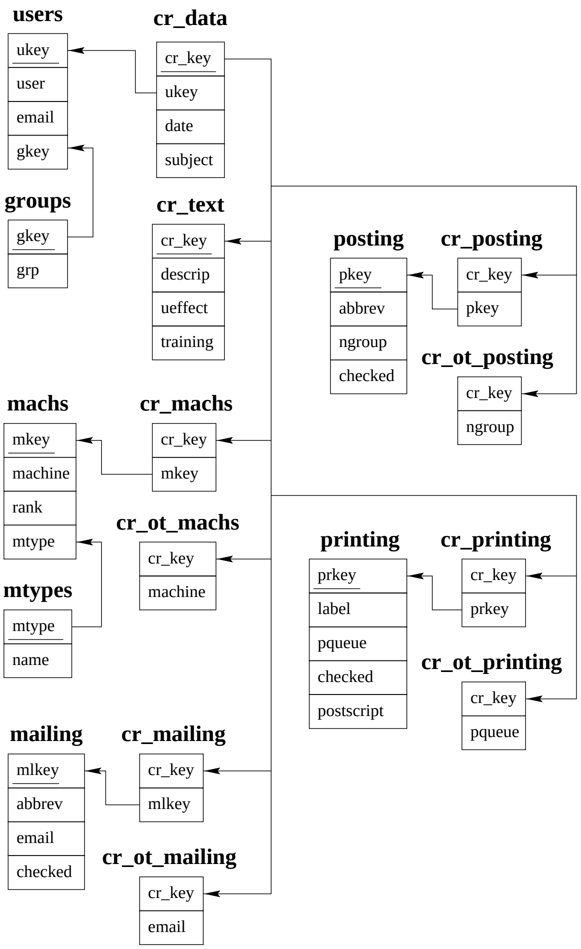


Figure 1 - System Change Report Database Schema

Integration

Now for the complex (yet most interesting) problem. You’ve got your web server hardware all set up, installed Apache, Perl, and MySQL or some other database and you’ve already set up your tables. What now? Well, we have to get your web server running Perl CGI scripts which will access your relational database. To do this, we will take advantage of the DBI (and DBD) and CGI modules. These modules, as well as a host of other Perl modules are available on CPAN (Comprehensive Perl Archive Network) sites [see Appendix]. I’m not going to go into great detail on what these packages can do for you. They all come with man pages and usually with extensive examples. I will, however, give the source code and a description of 2 simple Perl scripts that make use of them.

Simple Perl DBI Example

The following script is the equivalent to a DBI ’hello world’ program. It loads the MySQL DBD (database driver), uses it to connect to the "sccs" database on the machine that it is running (localhost), and uses the username "sprafka" and the password "Iam#1moM". Then it makes a simple SQL query, reports the number of rows, and prints out the query results as tab delimited text.

```
#!/local/bin/perl
use DBI;
my($dbh) = DBI->connect( "dbi:mysql:sccs:localhost", ’sprafka’, ’Iam#1moM’ );
my($sth) = $dbh->prepare("select * from participants");
$sth->execute();
print $sth->rows, " rows returned\n";
my(@row);
while (@row=$sth->fetchrow()) {
    print join("\t", @row), "\n";
}
$sth->finish();
$dbh->disconnect();
```

Simple Perl DBI & CGI Example

This next short script queries a database like the previous example, but it also creates an HTML form as well. As a part of this, we need more information about our database table. Let’s say that the ’participants’ table contains 2 fields or columns. First, *pkey*, which is a simple, unique (for this table), numeric identifier. And secondly, *pname*, which is a 40 byte (character) field containing the name of the participant. Here is the SQL statement for MySQL used to create this table:

```
CREATE TABLE participants (
  pkey tinyint(3) unsigned DEFAULT ’0’ NOT NULL auto_increment,
  pname char(40),
  PRIMARY KEY (pkey)
);
```

We will need to populate this table with some information, which is done with insert SQL commands. Once this is all done, we are ready to put a Perl script into place on the web server. Here is the code for our next example, which is followed by a detailed explaination.

```
#!/local/bin/perl
use DBI;
use CGI;
my($dbh) = DBI->connect( "dbi:mysql:sccs:localhost", ’sprafka’, ’Iam#1moM’ );
my($q) = new CGI;
print $q->header();
print $q->start_html(-title => "SCCS Registration");
print $q->h1("SCCS Registration"), "\n";
my($sth) = $dbh->prepare("select * from participants");
$sth->execute();
my(@row);
my(%participants);
while (@row=$sth->fetchrow()) {
    $participants{$row[0]}=$row[1];
}
$sth->finish();
$dbh->disconnect();
my(@pkeys) = keys(%participants);
@pkeys = sort { $a <=> $b } @pkeys;
print $q->start_form(-action => ’procsccs.cgi’, -method => ’post’);
print "Select your name:\n";
print $q->popup_menu(-name => ’reg_participants’,
          -values => [ @pkeys ],
          -labels => \%participants
         ), "\n";
print $q->br();
print $q->submit(-value => "Register Me"), "\n";
print $q->end_form;
print $q->end_html;
```

It is very important that you supply a username and password to the connect() call above. If you do not, you will most likely get an error when you try to access it through the web. Remember, most web servers run as user nobody on UNIX. If you do not set up permissions in your database to allow access from user nobody then you will have to supply a username and password in your CGI program. After we’ve gotten connected, we create an instantiation of a CGI object and place it in $q. Then we print out an HTML header (usually "Content-type: text/html"). Next, we call start_html(), which sets the title and prints the <HTML>, and <BODY> tags. Then we print out a nice banner in an <H1> tag. Notice that I’ve removed the line that tells us how many rows were returned from our SQL query, since we’re not really interested in that any more. Also notice that I’m now doing something different in the while() loop. I’m assigning the value of the first returned field, in this case *pkey*, making it the key in an associative array, and giving it the value of the second field, in this case *pname*. I’m doing this because I want to pass the value of *pkey* to the form processor and not *pname*. That way, the form processor can do a fast lookup on *pkey* and not a slow lookup on *pname*. Now that we know all of the participants and their keys, we make sure that the keys are in numerical order (we could have done this in the SQL statement), and assign those keys to an array, @pkeys. We’ll need this array later. Now we call start_form(), and pass it the usual form information, although it looks a little different. Now for the pop-up menu. After we print a short identifier, we call popup_menu(). The first argument, name, is obviously the name used in referencing this menu in the form processor. The second argument, values, is given a reference to an array that contains all of the possible values that will be sent to the form processor. The last argument, labels, is given a reference to an associative array that maps all of the possible values to labels that will appear in the actual menu that the user sees in the web form. We end the script with a submit button and the appropriate calls to print out ending tags like </FORM> and </BODY>.

In the end, the above CGI program produces the web form shown in the figure below [Fig 2]. But what has all this extra work gained us? Well, first off, we have at least a couple Perl scripts written that we can copy and modify in the future to suit our needs. But more importantly, we now have a dynamic web page. Let us say that as participants register themselves, they are removed from the table by the form processor and inserted into another table of registered participants. That way, the more people that register, the shorter our list in the web form becomes. Also, if new participants sign up, all we have to do is add their name to the participants table and our web form is automatically updated. What about output? We can easily generate lists of users from the tables of who has and who hasn’t registered yet. We pretty much did the former of these in our first example, we just need to add some HTML tags to it. The point isn’t so much that we *can* give this data to the end user, but that with a web form and CGI program, we can give them *exactly* what they want.

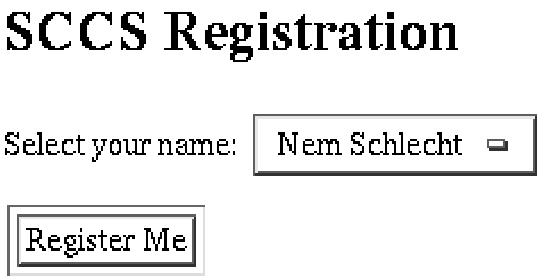


Figure 2 - Perl DBI/CGI Sample Output Form

I would like to note at this time that the above examples are pretty simple and are lacking proper error checking. The latest version of the DBI module automatically warns you of any errors, but when dealing with CGI programs running from the web, we often times will get very confusing results from an error and will most likely not see any error messages. However, these errors should show up in the error log file for your web server. That is the first place I go when my form looks incorrect. If you don't get any results whatsoever from your CGI program, the problem is most likely a syntax error in your Perl script. From the command line, you can ask perl to do a 'dry run' on your perl script and report any errors. You do this by running "perl -cw <*filename*>". This does not run your perl script, but rather checks for and reports any syntax errors or possible sources of problems. Lastly, you can similate the running of a CGI program from the command line by creating a file containing "form_variable=value" lines that correspond to the form variables and test values for your web form. You then run "perl *filename.cgi* < *form_vars_file*". You will then be able to see what parts of your CGI program succeeded and see the error message that would have been in the error log if it had been run by your web server.

Learning Curve

The learning curve in creating and maintaining these types of dynamic web documents is very similar to the programming/design curve for Object Oriented programming. There is an initial, high level of design and consideration for what you want to accomplish. A simple project, like the "ACM Election" example, given below, requires very little setup time. However, the "System Change Report" example required several weeks to fully plan and implement. In putting my statistics class to good use, I've observed that the Web/Database design/learning curve looks similar to a Weibull distribution, with $\alpha = 4$ and $\beta = 2$, as shown below [Fig 3].

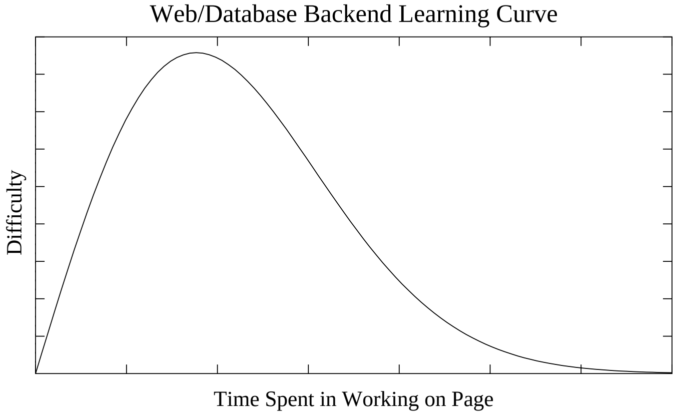


Figure 3 - Design/Learning Curve, Weibull distribution $\alpha = 4$, $\beta = 2$

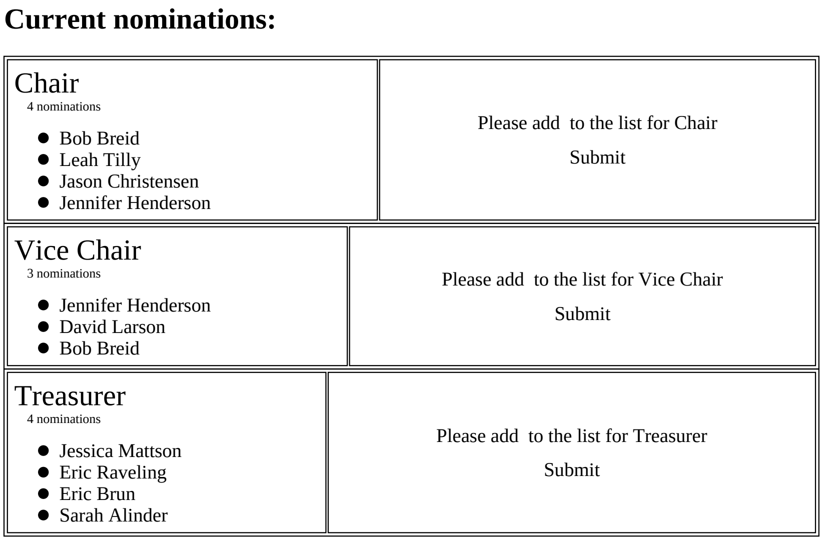
Yes, this curve looks very intimidating, but that should not discourage you. The reason I've telling you about this is because I do not want anybody to set their expectations too high. Creating these web pages is not an easy task, but with time and perseverance, it is achievable and there are many rewards.

Working/Example Models

Since any comprehensive examples would be very long and most probably very boring, I'm going to try to explain the key points in two systems that I have developed. Hopefully you will get some ideas or at the very least a decent understanding of what Web/Database integrated applications can do for you and your users.

Example - "A.C.M. Election"

Every year, my local A.C.M. Chapter holds elections for its Executive Council. This process includes having members that are non-eligible, or not interested in holding a position, volunteering to be on the Election Committee. That committee then collects nominations for individuals for a week or two. When nominations are no longer accepted, the committee must then start collecting absentee ballots. Then on election night, they actually count the votes. This year, I was on the Election Committee and decided to help streamline the process and also give users more anonymity by having them fill out a web form for any nominations [Fig 4] (text input fields have been replaced by underlined areas so that you can see them). Once I had this work done, I didn't need to do anything for awhile. When it came time to do absentee ballots, I created a table in my database for them, and another web form to collect the votes for me. For election night, I wrote up a Perl script to print out election ballots (although I had to count these by hand!). Again, as shown above in my 'Learning Curve' [Fig 3], I had a lot of initial work in setting up the tables and web pages, but once that was done I could add new features to the system and use the collected data in various ways very easily.



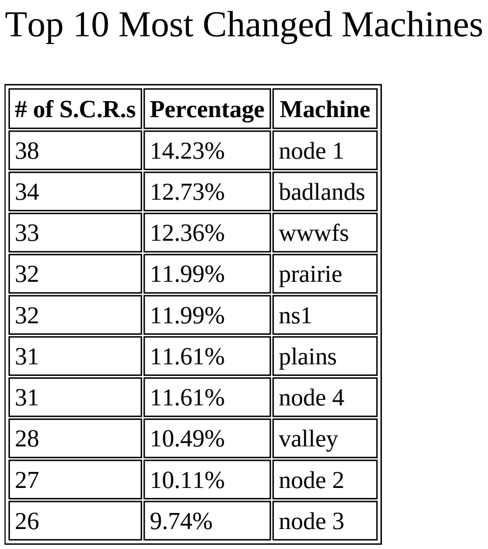
Current nominations:

Chair 4 nominations ● Bob Breid ● Leah Tilly ● Jason Christensen ● Jennifer Henderson	Please add to the list for Chair Submit
Vice Chair 3 nominations ● Jennifer Henderson ● David Larson ● Bob Breid	Please add to the list for Vice Chair Submit
Treasurer 4 nominations ● Jessica Mattson ● Eric Raveling ● Eric Brun ● Sarah Alinder	Please add to the list for Treasurer Submit

Figure 4 - A.C.M. Election Form

Example - "System Change Report"

In the group that I am a part of at I.T.S., we deal with the possibility of several different system administrators making changes to the same machines and not informing each other about it. For some time now, we have employed a "System Change Report" page, which in the past would merely e-mail certain staff members or Listserv lists, post the message to some newsgroups, or send the change report to a printer. Although this system was a start, we didn't archive any of the data in a central location. It was left up to each administrator to keep track and remember any changes any other administrator made to one of the machines that they watched as well. Often, mail was deleted or lost and there was little searching ability. I spent several months, working on and off, on developing and improving a new system. In the end, I came up with the current System Change Report or S.C.R. In the new S.C.R., not only is the initial page that comes up dynamic, but all the change reports are stored in the database as well. When I designed my database schema [Fig 1], I didn't even think about any applications or searching that I would want to do. I normalized my data, set up my tables, and wrote my CGI programs. After I got all that done, I decide that I wanted a page to do searches, a page that displayed any specific change report, and a page that gave out certain interesting statistics. The development of these pages took little time, and consisted mainly of just some SQL queries and formatting the output nicely. An example of one of the statistics tables is given below [Fig 5].



Top 10 Most Changed Machines

# of S.C.R.s	Percentage	Machine
38	14.23%	node 1
34	12.73%	badlands
33	12.36%	wwwfs
32	11.99%	prairie
32	11.99%	ns1
31	11.61%	plains
31	11.61%	node 4
28	10.49%	valley
27	10.11%	node 2
26	9.74%	node 3

Figure 5 - Sample Statistics Table from the S.C.R.

Where to go next?

Obviously, using a relational database, Perl, and the above mentioned modules isn’t for everybody. The idea is that one Perl/Database savvy person programs the templates and others fill in the data for them and make slight changes to already existing CGI programs. I think database technologies are really catching on. The advancements made in the freely available databases alone over the last year are quite impressive. PostgreSQL’s feature set includes views and transactions, both of which were unavailable on any free relational database a year ago. The MySQL programmers are constantly adding new features and enhancements. The Perl DBI interface has stabilized now into a virtually bug-free API. It’s a very exciting time for Web developers and administrators. I’m actually going to go out on a limb and predict that very soon now we will see a much tighter coupling between web servers and databases (I probably won’t escape the fact that most people that make predictions in the computer industry look like complete idiots in usually less than two years). Whether we can embed SQL statements right into our HTML documents with an <SQL> tag anytime soon or not, I don’t know. I do know that the amount of information that we are all handling will increase (isn’t it always that way!) and that we will have to find solutions to deal with it.

Appendix

Language Links

Official Perl Site: http://www.perl.com/
Official PHP/FI Site: http://php.iquest.net/
For a list of CPAN Sites: http://www.perl.com/CPAN

Web Server Links

Official Apache Site: http://www.apache.org/
Official NCSA HTTPd Site: http://hoohoo.ncsa.uiuc.edu/
Official Netscape Site: http://www.netscape.com/

Relational Database Links

Official MySQL site: http://www.tcx.se/
Official mSQL site: http://www.Hughes.com.au/
Official PostgreSQL site: http://www.postgresql.org/

Official DBI site: http://hermetica.com/technologia/DBI/
Official Oracle site: http://www.oracle.com/

Other Links

My papers: http://www.ndsu.nodak.edu/ndsu/nem/papers/
(Includes downloadable forms of the above samples and SQL tables given in this paper.)

Biography

Nem W. Schlecht is a UNIX Systems Administrator and Programmer for the North Dakota State University System. His specialties include Perl programming and database applications. His responsibilities include technical Web administration and systems administration for Silicon Graphics workstations. He also is a consultant for faculty, staff, and students on a wide range of UNIX related topics.

From On-Line to On-Task:
Communicative Technology in the First Year Classroom

Carolyn Schnell
College of University Studies
North Dakota State University
caschnel@plains.nodak.edu

Beginning Fall Semester 1997, using communictive technology effectively became a part of the General Education requirement at North Dakota State University. Two introductory courses devote a major component of the course to communicative technology. A required Freshman Seminar, during the second week of class, updates students on the use of on-line functions including e-mail and Web searches. The subsequent Fundamentals of Public Speaking class requires effective communication using Power Point technology during the last major student speech. This presentation will background each of these courses and describe the implementation and results of these technology units. Following the presentation, the session will be opened to group discussion.

From On-Line to On-Task:
Communicative Technology in the First Year Classroom

Carolyn Schnell
College of University Studies
North Dakota State University
caschnel@plains.nodak.edu

Introduction

The commitment of higher education to produce technologically literate graduates begins upon matriculation and continues until graduation. North Dakota State University President, Thomas Plough, stated in his 1997 State of the University Address, that "Our top priority will continue to be the preparation of our graduates for technological professionalism and leadership." He went on to state that "To achieve these kinds of educational outcomes requires learning opportunities both in and out of the classroom where theory and practice come together." The importance of a curriculum that enhances technological literacy and values the acquisition of skills necessary to use the computer as a tool are priorities for the institution. North Dakota State University has committed itself to the goal of technological literacy and implements a variety of programs to this end.

The goal of technological proficiency has come partly from graduate expectations voiced by employers. A reported 93.3% of employers expect e-mail experience, and 63.3% expect competency with online and Internet searching (Davis, 1997).

At North Dakota State University, the need for wider student access to updated technology began in earnest in the Fall of 1996. With student approval, a semester fee of $50 per 12 credits was instituted. With over $800,000 in revenue generated the first year and twice that expected over a two year period, the students and faculty alike have felt the impact of improved hardware and software.Areas impacted include but are not limited to: network access in the residence halls, union, and library; contribution to the NDSU web server and library database server; modem pool expansion; addition and improvement of workstations and cluster areas; additional hardware across campus including multimedia classroom equipment; and technical support including a multimedia center staffed by consultants 81 hours a week to provide WWW instructional support; and imaging and digitizing of media in addition to routine instruction and assistance. By the second semester of the l996-97 academic year, all classrooms on campus had been wired for networking capability, and wiring of residence hall rooms was completed by the following fall semester.

Students and faculty have access to and training with a wide variety of technological equipment. The Multimedia Center is staffed by two full-time employees and eleven student employees ready and willing to be of assistance to those who need help mastering the new equipment. Additional student help is hired to deliver and service the eighteen "carts" available for check-out and consisting of a proxima, a VCR, and in many cases, a document camera. Along with the carts, lap top computers may be checked out by students or faculty. In addition to upgraded computer hardware and software, various scanners, a CD-Rom burner, numerous zip drives, a video camera, two digital cameras, a parallel-port hard disk, and a DAT tape drive are available. The growing popularity of this equipment is evidenced by the report that the carts were reserved a total of 139 times over a ten week period between January and March of 1997. During a brief nine day period in January of 1998, carts were reserved a total of 113 times (N. Lilleberg, personal communication, January 2, 1998).

University commitments to technology are evidenced in part by two of the components of the General Education requirements, The First Year Experience Course and Fundamentals of Public Speaking. This paper will describe the technology component of these two courses and its current student impact.

Freshman Seminar Course

Background: Beginning Fall 1997, all new students entering North Dakota State University with 24 or fewer credits are required to complete University 199, "Skills for Academic Success," as part of the General Education requirement. This course is designed to ease the transition for new students at North Dakota State University. Class size for the sections discussed here is limited to 30 students in order to facilitate interpersonal interaction during the course. All classes are taught by instructors familiar with teaching at the college level and committed to the success and retention of college students.

Introduction of a required freshman seminar at NDSU was based on literature indicating increased success and retention for students completing such a course. A landmark study by Paul Fidler (1991) at the University of South Carolina indicated a significantly higher return rates to the sophomore year in 11 of the 16 years from 1973 to 1988 for students completing a first year seminar. Additional studies (Davis, 1992; House & Kuchynka, 1997) indicate increased GPA at the end of the first year for students who have completed a first year seminar. According to Gardner (1986), the first year experience is based on the concept that success during the first year provides the foundation on which the rest of the college experience is based. Within a supportive environment, "things happen by design, not by accident or spontaneity, i.e., those things that must happen if students are more likely to be successful" (Gardner, 1986, p. 267). It is the goal of NDSU to provide,during the early and vulnerable first few weeks a college, a supportive environment and some basic tools necessary for success during the next four years and beyond.

Description: The freshman seminar, "Skills for Academic Success," at NDSU, is intended to assist students in learning skills and techniques used by successful college students. In addition to introducing the students to campus resources and governance, topics include study techniques, time management, test taking, note taking, goal setting, wellness, stress management, and career orientation. Inherent to student success is the need for technology skills. Consequently, it is considered imperative that we introduce students to student technology skills and update the skills of those entering with some level of expertise.

Approximately 20% of the freshman seminaris designed to introduce students to technology on the campus. The goal of the technology unit, in concert with the technological goal of the institution, is to guarantee that all entering students are proficient in the technological areas of e-mail, use of the Internet, and library usage.A Fall 1997 University survey conducted by Information Technology Services indicated that, while 75% of these students have computers available at home or in their living quarters, only 49% report being "very comfortable" using e-mail, 37% report being "very comfortable" with search engines, and 26% report being "very comfortable" doing research on the Web.

Implementation: Given the importance of technological literacy, the technology unit was introduced during the third and fourth sessions of the course with the on-line library skills coming three weeks later. Scheduling and presentation were time and work intensive on the part of the instructors and staff. Thirty-two sections were offered to a total of 670 students in my unit with instruction to other units at a later date. Students were scheduled into computer clusters during their regular class hour with an effort being made to assure one computer per student. Instruction was delivered by professional trainers from the Information Technology Services. Students were required to bring a diskette to class, and instruction was projected visually via a proxima. Topics covered in addition to cluster information and Responsible Use Policies, included E-mail logins, Windows 95, the Pine mailer, and Netscape.

Results: Upon completion of the course, students were surveyed, and results were compared with survey results gathered the first day of the course. The first day of class, 50.6% of the students expressed agreement or strong agreement with the statement "I am comfortable using electronic mail" compared with 97.1% the last day. The first day of class, 49.4% of the students expressed disagreement or strong disagreement with the statement that "I am

comfortable using electronic mail" compared to 2.9% the last day. The first day of class, 65.6% of the students expressed agreement or strong agreement with the statement "I am able to search for a topic on the world wide web" compared with 95.8% the last day. The first day of class, 34.35% of the students expressed disagreement or strong disagreement with the statement that "I am able to search for a topic on the world wide web" compared to 4.2% the last day.
Results shown in Table 1 indicate significant improvement in all categories.

	Agree/Strongly Agree		Disagree/Strongly Disagree	
	#	%	#	%
I am comfortable using electronic mail.				
First Day	273	50.6%	267	49.4%
Last Day	563*	97.1%	17	2.9%
I am able to search for a topic on the world wide web.				
First Day	388	65.7%	203	34.4%
Last Day	570*	95.8%	25	4.2%

Chi-square $p \leq .001$

Table 1: Student Technology Self-Assessment Responses

Results of the student survey indicated a significant improvement of technology skills in both student comfort with the use of electronic mail and the ability to search for topics on the World Wide Web. In addition to the quantitative data obtained during the course assessment, comments from the students reflected a very positive response to this unit.

Public Speaking Course

Background: Acquisition of skills in the areas of e-mail, internet, and library usage during the fist few weeks insure that students have been introduced to technology, but the skills introduced are only the beginning. A logical second step is improvement in the ability to communicate using technology as a tool for accomplishing a task. Being involved in teaching both the introductory Freshman Seminar and classes within the Department of Communication led me to look for a way to integrate technology in an experiential manner into my communication classes. It was this goal that led me to initiate in my Public Speaking Class a pilot-course requiring the use of Power Point as a multi-media tool. I have taught Public Speaking classes for the last eight years and believe that there is legitimate and logical match in this area. This decision was supported by the advancing technological environment both on campus and beyond, the increasing university support previously mentioned, and a very positive reception by students in the classroom.

It may be premature to subscribe to Barry Brumett's (l991) vision of a future world.

> In one hundred years, when people laugh about the stories their great-grandparents tell of those who actually sat through ten-minute speeches, public communication will have taken the form of quips and quotes and wanton wiles, nods and becks and wreathed smiles. Public Video 101 will use a multimedia bank of teaching machines to present the "theory" of visual image construction, with a unit about midway through the course, earnestly taught to a somewhat bewildered class, on "the verbal."

However, it can be agreed that technology is leading and will lead to an explosive improvement in multi-media presentations. Anyone with a wordprocessor has the capability of producing documents with a quality reserved to a professional industry only a few years ago. Industry publications devote space to the development of effective presentation style (Bankerd, 1997). Any combination of words, graphics, sounds, or images can be set into motion or brought to life. In addition, our reliance on such technology impacts the way in which people learn "creating a shift in symbol-using, from a content-oriented cognitive system to a cognitive system based on form." (Johnson-Laird, cited in Chesebro, l995) In answer to the question "What do technological developments and changes in the mission and means of communication imply for future teaching and scholarship in Speech Communication?" Julia Wood and Richard Gregg respond that teachers of communication need to understand the increasing technological character of the world within which we live and its impact on the teaching profession (Wood and Greg, l995). According to Professor Laurence Alvarez, l996, "In a few years most of what is now thought of as instructional technology will be as much a part of the infrastructure as electricity and running water. Colleges that do not train their students in the proper use of technology are cheating them, and their students leave college ill equipped for the society of continual learning into which they move."

Indeed, our students, our classrooms, and our faculty cannot help but be impacted by technology. However, it is imperative that we realize multimedia in our classrooms is a means to an end and not an end in itself. Neither machines nor the multimedia presentations generated by them are as important as the students learning to use them, and the presentations are only as good as the creativity of the students who produce them. John Southard, Vice President for Computing Resources at Pratt Institute, stresses that technology cannot make up for poor management. It cannot define itself, it can not set directions, and it cannot lead. Most importantly, technology can't stop us from making mistakes (Southard, l990).

Once we accept the limits of technology, we are ready to incorporate its strengths into our classrooms. According to administrators of Information Technology at Virginia Commonwealth University (Gloster II, & Saltzberg, l995), technology can benefit learning in six areas:

* allows a student to take a more active role
* allows a teacher to express the content of a course in more than one format
* affects students by using techniques that reach various learning styles
* broadens the array of resources brought to a classroom and the student's workstation
* increases the opportunities for interactions between teachers and students and among students
* increases the productivity of those who support the learning environment.

Speech classes, by their nature, lend themselves to active student involvement, draw from extensive resources, promote student interaction, engage a variety of styles, and encourage productivity. Van Dusen (1997) in "The Virtual Campus" refers to the use of technology as a means of involving students with a variety of learning styles as well as specifically enhancing the lecture format. According to Van Dusen (1997, p. 41) "The interface, or point of interaction between the learner and the technology, becomes a window or a gateway to a variety of intellectually challenging activities." Obviously, presentations incorporating technology are a prescription for student growth and success.

Description: As part of General Education at North Dakota State University, all students are required to take Speech 110, Public Speaking. This class is generally taken within the first two years after matriculation. This class includes a speech of introduction, two informative speeches, an individual persuasive speech, a group persuasive speech, and an impromptu speech. During the course of the semester, students are introduced to theory, format, and presentation skills. Expectations include effective preparation and presentation of visual aids. The final speech of the semester culminates in a group presentation intended to foster group interaction, and to showcase skills achieved during the semester.

Implementation: I began incorporating multimedia use of *Power Point* into the final classroom presentation of Speech 110 in Fall 1996, and I repeated its use in Spring 1997. Based on that pilot program, I was convinced that technology should become part of a required expectation for our students who will be making their presentations to audiences of the 21st Century. According to Frank Newman, President of the Education Commission of the States (l990), "The average university in this country, in terms of its use of information technology in teaching, is substantially behind the typical elementary and secondary school." (Newman, l990) If universities are to close this gap, it is imperative that we begin quickly.

While today's students are becoming increasingly technologically literate, I was surprised that, during the l996/97 academic year, none of the students in my classes had experience with *Power Point* software. This technology became the medium for their final presentation. For anyone with minimal word processing skills, the software is easy to use with professionally designed templates prepared to guide the student through the design process. I chose *Power Point* by Microsoft due to my familiarity with the software and the fact that it is the industry leader most commonly used in business. (Beebe and Beebe, 1997) However, *Persuasion* by Adobe or *Freelance Graphics* by Lotus would serve the same purpose. I developed the following sequence supportive of this medium:

1. Midway through the semester, I conducted one of my classes using Power Point as the multi-media basis for lecture and discussion. In the course of a public speaking class, the instructor serves as a role model, and in this particular class I explained to the students the benefits and difficulties of relying on technology during presentation. One of my goals during the class was to engender enthusiasm for the use of *Power Point* as a means of enhancing communication.

2. For the last presentation, the class was divided into groups whose assignment it was to choose an issue relevant to higher education and to develop a persuasive presentation. Groups were allowed to choose their topic and encouraged to seek maximum impact with their choice of supporting data and presentation skills. Pascarella and Terenzini (1991) state that small groups encourage students to integrate knowledge into existing frames of reference and to make application of conceptual knowledge. Use of small groups allowed for a supportive environment and implemented the team-work approach encouraged by business today.

3. Training in the use of *Power Point* software was led by a professional from the technology center and one class session was devoted to instruction. I believe that as larger numbers of students are required to use this software it will be necessary to provide an alternate means of instruction. The classroom instructors may logically provide that instruction, or students may be able to learn from a tutorial. As current high school students enter college with better technology preparation, I believe that they will bring familiarity with *Power Point* or similar presentation software and it will not be necessary to spend a class period on instruction. However, until that time arrives, training appears to be a necessary part of this class.

4. During the following class period, students worked together in their groups. A computer cluster had been reserved for that purpose. Group support appeared to lessen the stress of learning a new program, and the students quickly incorporated the technology into their presentations. The work groups were enthusiastic. Students worked together on one computer with one disk and then copied it onto their own disk to continue working on their individual parts later.

5. Presentation day was unique for this set of speeches. As an instructor, I felt a lot more stress with the need to monitor the equipment before class in order to assure that it was all in working order. As the students gain more experience with technology in the next few years, I expect student comfort to increase and instructor involvement to abate. The greatest improvement in this set of speeches was the professional quality that the technology brought to the presentation. It was obvious from the student enthusiasm and pride that the use of *Power Point* generated a sense of confidence that had been lacking with poster-board visuals.

Results: The quality of the presentations is apparent in the video of the final speech. It is important here to note again that *Power Point* serves as a medium for communication. It is supportive of the presenter and not an end in itself. As Van Dusen (1997, p. 41) stated, “The technology itself, such as multimedia or hypertext, does not teach; it is, however, the vehicle for instruction set by the curriculum.” In this case, multimedia is not communication, but, when well-mastered, it becomes the vehicle for effective and enhanced communication.

Conclusion

The explosion of technology in our society is impacting higher education from before matriculation to after graduation. We expect our students to enter college with increased technology skills and both the students and their employers expect them to graduate with an ever-increasing level of technological sophistication. At times, there is concern that we lack the resources or the expertise to keep pace with such an explosion. However, by assuring, during the first two weeks of class that all students are comfortable with basic on-line skills and that by the end of the first year they are able to perform the task of effectively communicating in a professional manner, we have given them a foundation. From that foundation students can grow. Some will choose professions that require sophisticated technological knowledge and skills. Others will choose majors that require a minimal amount of technology usage.

The one thing of which we can all be assured is that our futures will require us to grow in our understanding and usage of technology. Unless we reach beyond where we are now, we will be left out of the wonder, excitement, and full potential that technology inspires.

References

Alvarez, L. R. (1996, May/June). Why Technology? [9 paragraphs]. Educom Review [On-line serial], 31(3). Available E-Mail: www.educom.edu/pub.

Bankerd, K., (1997). How to optimize projection technology: Using fonts, graphics, and color to maximize the effectiveness of your presentation. Syllabus, 11, (4). 32-35.

Beebe, S. A., & Beebe, S. J. (1997). Public Speaking An Audience-Centered Approach (pp. A-23). Needham Heights, MA: Allyn and Bacon.

Brummett, B. (1991). Rhetorical dimensions of popular culture. Tuscaloosa: University of Alabama Press.

Chesebro, J. W., (l995). Communication Technologies as Cognitive Systems. In J. T. Wood & R. B. Gregg (Eds.,) Toward the 21st Century The Future of Speech Communication (pp. 1-11). Cresskill, NJ: Hampton Press, Inc.

Davis, B. O., Jr. (1992). Freshman seminar: A broad spectrum of effectiveness. Journal of the Freshman Year Experience 4 (1). 79-94.

Gardner, J. N. (1986). “The freshman year experience. College and University 61 (4). 261-274.

Gloster II, A. S., & Saltzberg, S. A. (1995). Multimedia and asynchronous learning: Changing the role of academic computing. Proceedings of l995 CAUSE Annual Conference on Realizing the Potential of Information Resources: Information, Technology, and Services (pp. 5-8-1 - 5-8-10). Available E-Mail: www.cause.org.

House, J. D. And S. J. Kuchynka (1997). The effects of a freshmen orientation course on the achievement of health sciences students. Journal of College Student Development 38 (5). 540-541.

Newman, R. (1990). Technology on campus: An uneven Marriage [53 paragraphs] CAUSE/EFFECT [On-line serial], 13(1). Available E-Mail: www.cause.org.

Plough, T. R. (1997, Sept. 3). Acting Strategically Through a Culture of Achievement. State of the University Address, North Dakota State University, Fargo, ND.

Southard, J. (1990). What technology can do if we let it [9 paragraphs] CAUSE/EFFECT [On-line serial], 13(2). Available E-Mail: www.cause.org.

Van Dusen, G. C. (1997). The virtual campus: Technology and reform in Higher Eeucation. ASHE-ERIC Higher Education Report Volume 25, No. 5. Washington, D. C: The George Washington University, Graduate School of Education and Human Development.

Wood, J. T., & Gregg, R. B. (1995) The Future of the Field: Directing Scholarship & Teaching in the 21st Century. In J. T. Wood & R. B. Gregg (Eds.) Toward the 21st Century The Future of Speech Communication (pp. 1-11). Cresskill, NJ: Hampton Press, Inc.

Planet Oit: a Virtual Environment
and Educational Role-playing Game
to Teach the Geosciences

Brian M. Slator, Donald Schwert*, Bernhardt Saini-Eidukat*, Phil McClean**,
Jon Abel, John Bauer, Brian Gietzen, Nathan Green, Yongxin "George" Jia,
Tammy Kavli, Lucas Koehntop, Bhaskar Marthi, Vidyalatha Nagareddy,
Acey Olson, Kishore Peravali, Daniel Turany, Brad Vender, James Walsh

Computer Science Department
Geosciences Department*
Plant Sciences Department**
North Dakota State University
Fargo, ND 58105
contact: slator@badlands.nodak.edu

Abstract

The Geology Explorer project implements an educational game for teaching the Geosciences. This takes the form of a synthetic, virtual environment, Planet Oit, where students are given the means and the equipment to explore a planet as a Geologist would.

The game is designed to give students an authentic experience that will include elements of:

1. exploration of a spatially oriented virtual world;
2. practical, field oriented, expedition planning and decision making;
3. scientific problem solving (i.e. a "hands on" approach to the scientific method);

This paper describes a pedagogical architecture and an implemented application designed according to these principles. Students assume a role in a simulated environment and learn about real science by exploring in a goal-directed way and competing with other players. The game, which teaches principles of geology, is an implementation of a networked, multi-player, simulation-based, educational environment that illustrates the principles of learning by learning roles.

Introduction

People will invest extraordinary time and effort into learning how to play and win a game. Virtual role-playing environments can be a powerful mechanism for instruction, provided they are constructed such that learning how to play and win the game contributes to a player's understanding of real-world concepts and procedures. Simulated environments enable learners to assume roles in particular contexts and have meaningful, authentic experiences.

Designing educational games is an exercise in balancing trade-offs. Educational content should be foremost, but not by occluding playability and simple fun. Simulated situations should be familiar, or at least easily recognizable, but not at the cost of slavishly replicating all the tedious details of "real life". Experience in the simulated environment should be authentic, but not utterly predictable.

When these experiences are structured and arranged, even loosely, such that playing a role in the environment can illustrate the important concepts and procedures of the simulated domain, students are able to "learn by doing" [Dewey, 1900]. Meanwhile, the value of play in learning can hardly be over-stressed.

The Geology Explorer has been built to explore the following beliefs: (1)Educational technology should capitalize on the natural human propensity for role-playing, (2)students will be willing to assume roles if the environment makes it easy to do, and if the environment reinforces role-playing through careful crafting of the explicit tutorial components of the game, and (3) that educational software should be engaging, entertaining, attractive, interactive, and flexible: in short, game-like.

This paper describes a pedagogical architecture and an implemented application where students assume a role in a simulated environment and learn about real science by exploring in a goal-directed way and competing with other players. The game, which teaches principles of geology, is an implementation of a networked, multi-player, simulation-based, interactive, educational environment.

Platform Issues

Planet Oit is simulated on a MOO ("MUD, Object-Oriented", where MUD stands for "Multi-User Domain"). MUDs are typically text-based electronic meeting places where players build societies and fantasy environments, and interact within them [Curtis 1992]. Technically, a MUD is a multi-user database and messaging system. The basic components are "rooms" with "exits", "objects" and "players". MUDs support the object management and inter-player messaging that is required for multi-player games, and at the same time provide a programming language for writing the simulation and customizing the MUD.

The usual platform for operating a MUD or MOO is a machine running a Unix-based operating system, although there have always been alternatives, such as a MacIntosh server or a PC running Windows NT. Participants (usually referred to as players) connect by using Telnet or some other, more specialized, client program, which establishes a text-based session on the MOO.

Because the Geology Explorer project is intended to be a platform independent distance education system, the first client software for the project was a Telnet client developed in Java. This enabled connections from either MacIntosh, Microsoft Windows, or Linux X-Windows machines, using either Netscape or Internet Explorer browsers, although for technical reasons the Windows- and Linux-based browsing has been faster and better.

The client software to Planet Oit has been developed as different versions under the name GUMI (Graphical User-friendly MOO Interface). The first, a strictly text-based client, is called GUMI-bare. The more recent client development to implement more graphical functionality has gone forward under the name GUMI-game.

Design Issues

Research in active learning environments includes implementing "live" simulations for exploration and discovery that engage learners while treating them to a plausible synthetic experience. We have implemented the Geology Explorer as a synthetic environment using the freely available Xerox PARC LambdaMOO, which is a development environment for creating text-based virtual worlds, to simulate a portion of Planet Oit (very similar to Earth, and in the same orbit, but directly opposite the Sun). Students "land" on the planet to undertake an exploration exercise armed with tools and instruments implemented as LambdaMOO objects. They are given an authentic geosciences goal, e.g. to locate and report the position of potentially valuable mineral deposits. Accomplishing these goals will entail mastering several geoscience concepts and procedures, and will demonstrate student mastery of the material.

In many respects, Physical Geology is an ideal course for a role-based environment. Unlike many of the other sciences, Physical Geology is highly visual, with landscapes ranging from mountain tops to ocean floors, from arid badlands to intensely-leached tropical soils, from gently-flowing streams to violent volcanic eruptions.

However, it is obviously impractical to take large numbers of students into the field to experience first hand how a geologist makes decisions. However, students can do so in synthetic environments. Within this context, the student makes decisions similar to those of a geologist, using the tools and techniques of geoscience.

The first module mostly involves mineral exploration, where students are expected to plan an expedition, locate and assess potential mineral and ore deposits, and survive to report on it.

The first step was to develop a storyboard for the project which directed the development of the synthetic Planet Oit. A map was then developed to show the different environments on the planet (for example, Brown Dunes) and what will be encountered when the student travels southeast, the red beach, or south, the Lake region. A group of summer school students originally implemented multiple locations from which the geological expedition can begin. Geological tools were developed (such as streak plates, hammers, and Geiger counters), and the appearance and response of 40 minerals and 40 rocks to a series of interactions are described.

Once the layout and artifacts of Planet Oit had been implemented, the "rules of the game" were imposed. In particular, we have built an environment where students are transported to the planet surface and acquire a standard set of field instruments (a rock pick, a small bottle of hydrochloric acid, a magnet, and hand lens, a small glass plate and a streak plate). Students are issued an "electronic log book" to record their findings and, most importantly, are assigned an exploratory goal. These goals are intended to motivate the students to view their surroundings with a critical eye, as a geologist would. Goals are assigned from a principled set in order to leverage the role-based elements of the game.

The students can make their field observations, conduct small experiments, take note of the environment, and generally act like geologists as they work towards their goal of, for example, locating a Kimberlite deposit. A scoring system has been developed, so students can compete with each other and with themselves.

An on-line rock and mineral resource is being developed to allow students access to common reference materials. A simple tutorial browsing mechanism is also planned. Finally, a tracking mechanism has been implemented to follow students through the course of their explorations, in order to identify the way students are using the technology, and to implement software tutors.

Implementation Issues

The Geology Explorer project has gone through two phases: a text-based phase, during which Planet Oit was defined in terms of its Geoscience structure; and the current graphical phase, where the existing functionality has been visually enhanced with landscapes and other images.

Textual Implementation

Planet Oit development has been accomplished by creating objects in the LambdaMOO environment and implementing methods (verbs) on those objects in order to simulate an authentic exploration and problem-solving experience. To accomplish this, the following classes of objects have been implemented:

- ° objects representing geological "spaces" (mountains, caves, buttes and the like -- these are implemented as instances of the LambdaMOO "$room" object);
- ° objects representing geological "entities" (outcrops, minerals, streams, and so forth -- these are implemented as instances of the LambdaMOO "$thing" object), and programmed to act and react authentically.
- ° objects representing the "tools" and "instruments" of Geologists (both "field instruments" like a compass, a rock pick and an acid bottle, and "laboratory instruments", spectrometers etc., to perform complex analyses -- these are also be implemented as LambdaMOO "things")
- ° objects for representing game players (these are special instances of the LambdaMOO "$player" type)
- ° objects representing on-line tutors (these are software agents implemented as another special instance of the "$thing" type)

Locations

Planet Oit is built out of an "entryway room" which represents the expedition's landing and staging area, with exits leading toward each of the compass directions. There are seven (7) main areas adjacent to the Planet Oit entryway:

1. To the north is a glistening, azure, ocean seashore.
2. To the northwest you see a sparkling inland lake

3. To the west is a majestic range of chiseled mountains
4. To the southwest you see a vast expanse of open prairie
5. To the south is a blistering desert
6. To the east is the soft outline of a mountain range, and
7. To the northeast you see a broad area of rolling hills and valleys

On Planet Oit, every exit must have a direction as one of its names (e.g. "East"), and a letter-direction name (e.g.. "e"), and a room-direction name (e.g. "cave"). Therefore, a player in the Old Mountains can type "n", or "north", or "cave" and get to the "cave with stalactites".

Rocks and Minerals

Rocks and minerals on Oit are implemented as objects of type $thing. Each object was further imbued with properties for:

1. what kind of rock it is (igneous, sedimentary, or metamorphic)
2. a detailed description
3. capable of spawning children
4. immovable so it can't be carried off (however, the children of a rock must be movable, so they can be carried to the laboratory for analysis.
5. values to the .odor, .flavor, and .texture properties if they differ from the defaults
6. properties and values appropriate to a specific rock or mineral:
    a. .density
    b. .height, .weight, .depth (in integer inches)
    c. .color (a string)
    d. .luster (a string)
    e. .magnetic (a 0 or 1)
    f. .hardness (float value in the range 0.0 to 10.0)
    g. .minerals (a list of (proportion mineral) pairs)

Once objects for rocks and minerals were defined, verbs were written to describe each rock's behavior when they react with the Geologist's instruments. For example, verbs to react to "hitting" (with a hammer or rock pick), and for "pouring" (a 10% solution of Hydrochloric Acid), were specified as follows:

hit_by

The player will say:

hit rock-name with instrument-name

The instrument defines a "hit" verb, which produces a message and then calls a verb as follows:

rock-name:hit_by(instrument-name)

The rock or mineral defines the appropriate hit_by behavior to handle the following cases:

hit_by hammer / rock_pick => results in:
hit_by jack-hammer => results in:
hit_by derrick-drill => results in:

chip : a message that chips are flying
split : a message, and create a movable child (appropriately sized, etc.) of the rock
destroy: a message and recycle the rock
nothing: a message that nothing happened.

hit_by OTHER

What happens when other things (i.e. the compass or the gravimeter) are used to hit a rock?

poured_on_by

The player will say:

pour liquid-name on rock-name

The liquid defines a "pour" verb, which produces a message and then calls a verb as follows:

rock-name:poured_on_by(liquid-name)

The rock or mineral defines the appropriate poured_on_by behavior to handle the following cases:

poured_on_by acid-bottle or
poured_on_by water-bottle/canteen or
poured_on_by aqua-regia or
poured_on_by heavy-liquids => results in one of:
foaming: a message about foaming behavior
nothing: a message that nothing happened.

poured_on_by OTHER

What happens when other things (i.e. the compass or the gravimeter) are used to pour on a rock?

Similar protocols have been developed for scratched_by, viewed_by, touched_by, measured_by, irradiated_by, and processed_by.

Instruments and Tools

To create instruments it was necessary to do the following:

1. find out what kind of instrument it is: a laboratory instrument, a field instrument, or some other kind.
2. make field instruments fertile because everyone who plays will need one.
3. locate laboratory instruments in the Laboratory, and they should be immovable.
4. create an instrument object
5. describe the instrument
6. assign values to the .odor, .flavor, and .texture properties if they differ from the defaults
7. assign values to the .height, .width, and .depth properties, (integers representing inches), and the .weight property (a list of two integers for pounds and ounces: i.e. {5, 0}
8. write verbs on the instruments, with appropriate error checks, to describe its behavior when reacting with rocks and minerals, or other elements of the environment, as necessary.

   These verbs are mostly short and consisting of two things:
   1) messages describing the actions of an instrument in terms of sight and sound; and
   2) a "message" sent to, or a verb invoked on, the object of the instrument's action. For example, the hockey stick makes a "whooshing noise". Then it calls the hit_by verb on the relevant object.

Graphical Implementation

A key element of the exploratory game idea is the notion of a spatially oriented synthetic environment where learners explore and discover. The spatial metaphor maps a domain (and, consequently, its interface) onto the basic spatial elements on Oit. The Geology Explorer accomplishes this using client software written in Java that is a viewport into the MOO running the game server. In it, objects are represented by graphical elements that can be manipulated in a way that makes sense to the domain.

One major shortcoming of MOOs and MUDs, however, is their low-tech communication system: text. The Geology Explorer now supplies a graphical user interface layered on top of the networked multi-user database and messaging system that MUDs provide.

To accomplish this, a Java client and communication protocol were developed. These support the viewport which is an accurate and consistent representation of the data on the game server. Changes in the server are reflected in the viewport, and manipulations of the viewport change the state of the server. The viewport on the client machines is a view in a window which displays pictures that represent MOO objects such as exits, objects, and other players.

The viewport is responsible for:

1. Storing the current room information (identification)
2. Storing a list of objects in that room
3. Notifying those objects when their state changes
4. Notifying the server when the user manipulates the objects

The viewport is used mostly for protocol between the server and the objects in the room, and also between the objects and the platform's user interface routines. Objects in the room are stored as viewlogos. Viewlogos are responsible for:

1. Storing the object name and id
2. Storing the object's position
3. Retrieving and displaying the object's image
4. Responding to clicks, double-clicks, dragging, and drop requests.
5. Responding to state changes by updating the object's image
6. Possibly support animation

There are different subclasses of viewlogo for different classes of objects in the MOO. The three main subclasses are Object, Exit, and Player.

Tutoring Issues

On Planet Oit, tutoring is done through non-intrusive but proactive software agents. Agents monitor student actions and "visit" a student when the need arises. Tutors give advice, but they do not mandate or insist on student actions, nor do they block or prevent student actions.

There are currently two types of tutoring agents in the game, but a third type is planned:

The Equipment Tutor

The equipment tutor has been implemented to detect when a student has failed to acquire equipment necessary to achieving their goals The equipment tutor is mainly called by the *purchase* verb (which is how instruments and tools are acquired. The tutor checks whether the student has the instruments needed to satisfy their goals. If not, the tutor remediates on that topic (i.e. the need to buy instruments that serve to satisfy goals). For example, an acid bottle is necessary to identify limestone. If the student has a limestone goal, but has no acid, the student cannot possibly achieve the goal.

Future plans may call for the tutor to check whether the instrument purchased can be used to satisfy any of the player's goals. If not, the tutor may decide to remediate on that topic (i.e. buying instruments that serve no obvious purpose)

The Exploration Tutor

The exploration tutor has been implemented to detect when a student has overlooked a goal in their travels. The exploration tutor is called by the exit(s) from each of the locations (rooms) on Planet Oit. The tutor checks whether the student is leaving a room that might satisfy a goal; i.e. if their goal is to locate Kimberlite, and there is Kimberlite in the room they are leaving, the tutor visits the player to inform them.

Future plans call for the tutor to decide whether to remediate on that topic. This remediation could be done on a room-by-room basis (easiest), or it could be done on a region-by-region basis (harder); or on a hot-cold basis (i.e. if the player is moving farther away from some distant goal). Or the tutor may decide not to remediate at all.

The Science Tutor

The science tutor will be implemented to detect when a student makes a mistake in identifying rocks and minerals. This tutor will activate to tell a student a wrong guess has been made and why (i.e. what evidence they are lacking), or to tell a student making a correct guess that insufficient evidence has been gathered (i.e. a lucky guess) .

The science tutor is called by the *report* verb (which is how players score points: by showing they have achieved goals). The tutor checks the player's history property (which is where the tracking information is stored on each player), and determines which of the following cases are relevant:

1. (wrong tests) the player has "guessed" incorrectly and the player's history property indicates they have not conducted the necessary tests to identify the rock/mineral in question
2. (wrong answer) the player has "guessed" incorrectly and the player's history property indicates they have conducted the necessary tests to identify the rock/mineral in question
3. (lucky guess) the player has "guessed" correctly but the player's history property indicates they have not conducted the necessary tests to identify the rock/mineral in question
4. (good work) the player has "guessed" correctly and has conducted the necessary tests to identify the rock/mineral in question.

Depending on the results of the reporting analysis, the tutor may decide to remediate on the spot, or may decide to defer remediation until the player begins to show a pattern of behavior. The Science Tutor will work from knowledge of the rocks and minerals, and knowledge of the "experiments" needed to confirm or deny the identity of a rock or mineral.

The system will encode the necessary and sufficient experiments for each rock and mineral, as well as their expected results. The system will check these facts against the student's history property whenever the student "guesses" a deposit's identity The system will remediate, as appropriate, according to the four cases listed above.

Assessment Issues

Developing methods for the assessment of student learning are a central element of the research. Briefly, the assessment goal is to determine the benefit to students derived from their "learn by doing" experience on Planet Oit.

The assessment strategy rejects the notion of standardized multiple choice tests as an adequate instrument in this pedagogical context. While there are, indeed, facts and concepts acquired in the course of exploration, which are neatly packageable and testable with objective instruments, the effect on student learning in that arena will not be significant, nor would we expect it to be.

Therefore, the assessment protocol designed for the Geology Explorer is a subjective one that seeks to measure how student thinking has improved. To do this, players are given a subjective, narrative based survey where they are told short problem solving stories and asked to record their impressions and any questions that occur to them. These surveys are analyzed for the presence of what could be considered "important" Geological or problem solving concepts or procedures.

Then, after the players have experienced an extended exploration of Planet Oit, they are given a similar post-test survey with different but analogous problem solving scenarios, and asked again to record their questions and impressions. These documents are then compared with the pre-test versions and evidence of improved performance is looked for. If players exhibit a better understanding of the problem solving scenarios, this creates the clear implication that they have learned from the game. We plan an online assessment system to automate this process.

Related Work

Overwhelmingly, the most common approach to implementing synthetic multi-user environments is the text-based MUD: the multi-user, text-based, networked computing environments that are mostly for "gaming". MUDs, or Multi-User Dungeons, are an evolution of computer chat utilities and bulletin boards combined with the popularity of adventure role-playing systems, such as Dungeons and Dragons. They are environments which one can connect to from an Internet terminal, then interact in text with objects, places, and other players within a game-like setting [Carlstrom 1992].

The Social Virtual Reality project at Xerox PARC has extended MUD technology for use in non-recreational settings. Their goal is to keep the strength of MUDs --- shared computing with a powerful real-world metaphor --- while correcting their shortcomings by adding audio, video, and interactive windows. They have built two specific prototypes: Astro-VR, for use by the professional astronomy community, and Jupiter, for use by researchers within Xerox. [Curtis and Nichols, 1993]

In a recent search of the World Wide Web it was clear that MOOs for different ability levels are becoming a reality. Amy Bruckman, a doctoral student at the Massachusetts Institute of Technology has built a programming language to make it simpler for children to construct objects and participate in MOOs [Bruckman, 1993]. She has combined construction and community in the hope of creating a constructionist learning culture in her MOOse-Crossing MOO.

The Donut MOO in Stark County, Ohio addresses the needs of K-12 students. Students build the MOO by creating "textually anchored virtual reality spaces." Although the site is open to all students, many are older.[Suzie 1995]

MOOs have shown their importance in elementary schools. Two in particular, MariMuse, and MicroMUSE have been geared so that elementary school students can participate full-time. One notable success has been on underachieving students who had left school. These students reportedly became involved, started to form friendships, and began to take a greater interest in school.[Poirer 1995]

In some learning environments, the virtual reality MOO connected to a network is already here. At the United States army base and training facility in the Mojave desert, there is a virtual reality multi-user computer simulation that can be linked to other military bases around the world. Using topographical resources and mapping facilities, the entire Mojave desert has been re-created digitally. Soldiers from all over the world can participate in the same wargame scenario.

Other examples of virtual reality MOOs, sometimes called "multi-user computer simulations,", are being implemented in a virtual physics classroom being developed at NASA, and with interactive programs run the Loma Linda Medical center in southern California [Mclellan 1994]. Mclellan [1994] cites some early conclusions about the VR MOO experience with other entertainment games such as "Battletech".

Mineral Venture is a recently developed software environment that simulates business-oriented mineral exploration from a technical and economic perspective. This is not a multi-user spatially oriented exploration system, but rather a simulation intended to pose planning and resource management problems that geologists routinely face.

SELL! is a multi-player, networked game that teaches basic marketing and micro-economic concepts. Players are immersed in a simulated environment where they are expected to save a failing retail outlet. The tools of the retail trade (hiring, advertising, ordering, pricing) are made available, and the underlying simulation is crafted to respond to game play in plausible ways. Throughout the course of a game, players have the opportunity to consult real world entrepreneurs, advertising executives and economists for guidance as they attempt to build up the net worth and market presence of their simulated businesses [Slator and Chaput, 1996; Hooker and Slator, 1996].

Programming Land MOO [Hill and Slator, 1998], at Valley City State University is being developed as an adjunct to programming classes. The MOO contains material that parallels an introduction to programming in C++, Java, and Basic. The course is modelled as a Virtual Lecture built using the active museum metaphor. Students embark on a self-paced exploration of the museum, where active exits act as tutors who advise whether a student is likely prepared for the material within.

Conclusion

The text-based Geology Explorer is in a fairly advanced prototyping phase, and preliminary pilot testing is planned for the near future. From this we hope to move forward on four fronts: 1) the assessment of student uses of educational games; 2) the development of a theoretical construct for explaining student use; 3) the prototype development of an experimental game; and 4) the design of a suite of software tools for continuing the development cycle.

Experiments will be conducted so as to answer particular research questions concerning 1) the effectiveness of these educational environments in terms of student learning and technology use, and 2) the effectiveness of our approach to game design and development in terms of code re-use and tool development.

The Geology Explorer will become an interactive multi-media (graphical) educational game for teaching Geoscience in a role-based, goal-oriented, and learn-by-doing way. With that we will study several issues connected with highly graphical and highly interactive learning technologies with a view towards developing effective teaching systems and efficient methods of implementation.

References

Bruckman, Amy (1993) MOOse-Crossing Thesis Proposal, Cambridge: MIT.

Carlstrom, Eva-Lise (1992). BETTER LIVING THROUGH LANGUAGE: The Communicative Implications of a Text-Only Virtual Environment. Student Paper, Grinnell College.

Curtis, Pavel (1992). Mudding: Social Phenomena in Text-Based Virtual Realities. *Proceedings of the conference on Directions and Implications of Advanced Computing* (sponsored by Computer Professionals for Social Responsibility)

Curtis, Pavel and David Nichols (1993) MUDs Grow Up: Social Virtual Reality in the Real World, *Third International Conference on Cyberspace*, May.

Dewey, J. (1900). *The School and Society*. Chicago, IL: The University of Chicago Press.

Hill, Curt and Brian M. Slator (1998) Virtual Lecture, Virtual Laboratory, or Virtual Lesson. Proceedings of SCCS '98. Fargo-Moorhead. April

Hooker, B. and B. Slator, B (1996) A Model of Consumer Decision Making for a Mud-based Game. ITS'96 Workshop on Simulation-Based Learning Technology. Montreal, June.

McLellan, Hilary (1994), Virtual Reality Goes to School, *Computers in Schools,* Vol. 9, No. 4

Poirer, Joseph R. (1995), "Interactive Multiuser Realities: MUDS, MOOS, MUCKS, and MUSHes, *The Internet Unleashed.* (Indianapolis: Sam's Publishing), pp. 1126-1127.

Slator, Brian M. and Harold "Cliff" Chaput (1996). Learning by Learning Roles: a virtual role-playing environment for tutoring. *Proceedings of the Third International Conference on Intelligent Tutoring Systems (ITS'96).* Montreal: Springer-Verlag, June 12-14, pp. 668-676. (Lecture Notes in Computer Science, edited by C. Frasson, G. Gauthier, A. Lesgold).

Suzie, John. (1994) Donut: Starknet Campus of the Future, *The Journal Of Virtual Reality in Education* Volume 1, No.1, pp. 46-47.

Acknowledgments

The following people were involved with the Geology Explorer as a class project in its earliest stages: Scott Gordon Aagard, Elson Abraham, Brian Raymond Allrich, Tonia Marie Brezina, Michele Lea Chown, Murali Dhandapani, Michael Glen Dunkle, Brent Alan Ellingson, Chad Justin Elliott, Bryan James Fugere, Syed Habibullah, Jeffrey Allen Haugen, Robert Jerome

Hoffman, Jim Sam John, Beau James Douglas Kautzman, Shane Nelson Kullman, Jason T. Lagge, Debra Ann Meyers, Rupa Mitra, Prashanth Mylvarabatla, Curtis Wayne Ophoven, Xiaofeng Pan, Mingbo Qin, Sylvia Salas, Darin Jeffrey Schmitz, Samuel E Silverthorn, David Chadwick Teigland, Daniel Douglas Ward, Yue Yun.

Additional thanks to Mark Tinguely, who saved our world when its universe imploded, and to Dave Schmidt for the name: Planet Oit.

This project was supported in part by a Grant from the NDSU PPRC to the NDSU WWWIC administered by Dr. Paul Juell and Dr. Phil McClean, by NDSU Grant-in-Aid #1109, and by an emergency equipment grant from ND-EPSCOR, Dr. Phil Boudjouk, director.

Appendix 1: a Transcript from Planet Oit

<the player logs on as demoplayer>

```
Welcome to the NDSU/CS Geology Explorer MOO

We are implementing Planet Oit for you to explore
in order to learn something about geology.
For more information on this project, point your web browser to:
     http://www.cs.ndsu.nodak.edu/~slator/
For an online user's manual, point your web browser to:
     http://www.cs.ndsu.nodak.edu/~slator/html/PLANETX/planet-x-userman.html

PLEASE JUST CREATE *ONE* CHARACTER FOR YOURSELF (TO USE EVERY TIME)

     Problems? email slator@badlands.nodak.edu

*** Connected ***
Planet-X
You are standing in the middle of the expedition staging area on the surface
of Planet Oit. You see a spaceship on the ground with its bay doors open,
exposing a laboratory within. A glorious panorama spreads out on all sides.
     There is a Way Station here
*    To the North is a Seashore
*    To the Northeast are the Hills
*    To the East are the Old Mountains
*    To the Southeast is a Laboratory
*    To the South is a Desert
*    To the Southwest are the Plains
*    To the West are the Young Mountains
*    To the Northwest is a Lake
Obvious exits are: Down Station North Northeast East Southeast South Southwest
West Northwest

As a player in the Geology Explorer
   You have reached no goals yet, and you are playing at the beginner level.
   You have a general goal: to 'report' all the rocks and minerals you find
       and a specific goal: to 'report' the location of Native Copper.
   Native Copper is a reddish metallic mineral.
   Native Copper is used in electronics.
     You can expect to find Native Copper in Desert (#147)
     In particular, you should look in Plateau with Fissure (#350).
To see this message again, type '@showgoal' or 'showgoal'
```

```
=> LOOK ME
demoplayer
You see a player who should type '@describe me as ...'.
It is awake and looks alert.

=> @DESCRIBE ME AS "AN INTREPID ADVENTURER BOLDLY GOING WHERE NO ONE HAS GONE
      BEFORE
Description set.

=> LOOK ME
demoplayer
an intrepid adventurer boldly going where no one has gone before
It is awake and looks alert.

=> HELP ME
The Geology Explorer is a game and a "learn by doing" environment
  for teaching the Scientific Method and deductive reasoning
  in a synthetic exploratory virtual world.
To play the game, you identify rocks and minerals on Planet Oit
  and you score points for each one.
Normal game play will include the following steps:
    1. You log on to the game using a name and a password
         this transports you to the surface of Planet Oit
    2. You are assigned a general goal and a specific goal
    3. You acquire the tools and INSTRUMENTS you need
         from the STORE on your space ship
    4. You explore the planet, seeking your goals
    5. You REPORT the rocks and mineral to score points
Hints:
    You will find the "help" system very useful
      See especially: "help instrument" and "help <verb>"
        (where <verb> is one of hit, pour, scratch, streak
          touch, view, taste, or smell)
    If a tutor visits you, pay attention
    Watch out for wild animals
Welcome to Planet Oit!

=> HELP NATIVE COPPER
Native Copper (mineral)
  color: reddish
  heft: very heavy
  texture: smooth
  acid reactive: 0
  streak: Faint, copper-colored
  luster: Metallic
  hardness: from 2.5 to 3.0
a soft reddish metal that is a simple substance (ELEMENT), easily shaped, and
 allows heat and electricity to pass through it easily (LDOCE, 1978)

=> SOUTHEAST
Laboratory
You are standing in the laboratory of a spaceship, you are surrounded by many
 instruments.
A sign on the wall says,
    'You can buy field instruments to the East.'
     The sign continues:
        'You can use these lab instruments by bringing samples back here.'
*    To the East is a Shopping Center
*    To the South is a Rock Museum
*    To the Northwest is  Planet-X
Obvious exits are: East South Northwest
You see Thin Section Machine (#365), Bunsen Burner (#576), Rock Powdering
 Machine (#603), X-ray Fluorescense Spectrometer (#590), Rock Crusher (#753),
 Laboratory Densitometer (#714), Mass Spectrometer (#763), Aqua Regia (#566),
 X-ray Diffractometer (#299), Heavy Liquids (#604), Magnetometer (#578),
 Atomic Absorbtion Spectrometer (#592), and Derrick Drill (#586) here.

=> EAST
Crazy Eddie's Geology Equipment Emporium
Crazy Eddie's is the place to go to fill all your geology field equipment
needs.  Aisle after aisle of merchandise makes this place into a maze.
You could easily get lost in here without assistance.

A large sign on the far wall reads:

         Trouble finding what you're looking for?
                Look it up in our catalog!

         (Never mind the cost -- everything is free)

Obvious exits are: West
You see Crazy Eddie and Catalog here.
Crazy Eddie says, "Hello, demoplayer.  Welcome to Crazy Eddie's Geology
      Equipment Emporium!
Crazy Eddie says, "I've got the best deals on Planet X!"
Crazy Eddie asks, "Can I get you anything?"

=> LOOK CATALOG
                **** Crazy Eddie's Mail Order Catalog ****

           Item            Price   |             Item          Price
Acid Bottle (#597)       $    1.00 | Light Meter (#556)       $25.00
Altimeter (#675)         $  100.00 | Magnet (#580)            $1.00
Anemometer (#568)        $   30.00 | Microscope (#612)        $25.00
Barometer (#410)         $   50.00 | Rock Pick/Hammer (#370)  $40.00
Black Light (#625)       $   30.00 | Sextant (#554)           $100.00
Compass (#561)           $  200.00 | Sonar (#593)             $200.00
Flowmeter (#598)         $  100.00 | Spirit Level (#335)      $5.00
Geiger Counter (#751)    $  178.00 | Streak Plate (#542)      $0.50
Glass Plate (#583)       $    0.50 | Tape Measure (#626)      $10.00
Goniometer (#613)        $    1.00 | Thermometer (#223)       $5.00
Gravimeter (#708)        $10000.00 | Transit (#364)           $10.00
Jack Hammer (#368)       $  300.00 | Water Level Gage (#615)  $5.00

=> HELP HARDNESS
hardness: Mineral's resistance to scratching on a smooth surface. Mohs scale
 of relative hardness consists of 10 minerals, each scratching all those below
 it in scale and being scratched by all those above it: 1) talc, 2) gypsum, 3)
 calcite, 4) fluorite, 5) apatite, 6) orthoclase, 7) quartz, 8) topaz, 9)
 corundum, 10) diamond
Crazy Eddie twiddles his thumbs.

=> SAY SELL ME A ROCK PICK
You say, "sell me a rock pick"
Crazy Eddie quickly disappears for a moment.
He reappears carrying a Rock Pick/Hammer (#2757), which he gives to you.
You have accumulated $40.00 in charges.
```

```
=> SAY SELL ME A GLASS PLATE
You say, "sell me a glass plate"
Crazy Eddie quickly disappears for a moment.
He reappears carrying a Glass Plate (#2758), which he gives to you.
You have accumulated $40.50 in charges.

      <the player travels back to the Plateau with Fissure>

Plateau with Fissure
You are in an exotic place with a great view of the surroundings. This
 plateau, sometimes called tableland, is a large level area raised above the
 adjacent land.
There is a fissure here (a narrow opening or crack) -- it is dark within, you
 cannot see inside. You see wild flowers and green trees here. A condor
 circles lazily overhead.
*    To the South is a White Dune
*    To the Southwest is a Desert
*    To the West is a Cinder Cone
Obvious exits are: South Southwest West
You see dark green-grey coarse-grained outcrop (#1902), light green granular
 medium-grained outcrop (#1905), reddish metallic crystal (#2008), black
 opaque submetallic crystal (#2019), dark brown opaque submetallic crystal
 (#2021), light gray banded granular outcrop (#2046), dark greenish-black
 coarse-grained outcrop (#2077), and dark gray fine-grained outcrop (#2083)
 here.

=>WEST
Cinder Cone
You are standing at the base of conical hill formed by the accumulation of
 cinders around a volcanic vent. The landscape here is brown and black, there
 is very little in terms of green growing things. The air smells faintly of
 sulphur.
*    To the East is a Plateau with Fissure
*    To the Southeast is a Desert
*    To the South is a Reddish Butte
*    To the Southwest is a Cave
Obvious exits are: East Southeast South Southwest
You see dull reddish porous medium-grained outcrop (#1923), dark gray
 fine-grained outcrop (#2082), black smooth glassy block (#2087), and grayish
 white rough glassy porous block (#2100) here.

      <the player is visited by the tutor>

A clap of thunder splits the air, and a tutor appears in your midst
  The tutor appears to be speaking to demoplayer
TUTOR: You just left Plateau with Fissure containing your goal: Native Copper
The TUTOR bends at the waist and disappears in a puff of smoke

      <the player returns to the Plateau with Fissure>

=> HIT REDDISH CRYSTAL
You hit reddish metallic crystal (#2008) with your hands. Nothing happens

=> SCRATCH REDDISH CRYSTAL WITH GLASS PLATE
reddish metallic crystal (#2008) does not scratch Glass Plate (#2758)
    Note: reddish metallic crystal (#2008) is soft enough to bend. Is is very
       malleable.
```

```
=> REPORT REDDISH CRYSTAL AS NATIVE COPPER
Checking: reddish metallic crystal (#2008) against Native Copper (#673)
You are right! reddish metallic crystal (#2008) IS native copper
You score 10 points!
You have satisfied your primary goal! You score 500 points.
```

<the player is assigned a new goal, and continues the game>

University of North Dakota
Academic Affairs Web Initiative Pilot

Kathy Smart, Ed.D.
Steve Pottenger, B.S.
Vicki Wessman-Downey, Ph.D.
Bette Olson, M.S.
University of North Dakota

Introduction

The University of North Dakota Center for Instructional & Learning Technologies is currently piloting the Academic Affairs Web Initiative (AAWI) for faculty who are interested in exploring technology but do not have the time or resources to start from ground zero. AAWI is a customized tool designed for faculty to generate individual course web sites following a one hour orientation session. AAWI is a database-driven system that uses a web-based interface to provide individual web sites within a structure of colleges and associated departments. This new model for organizing supplemental course materials on the web has the potential to provide a cost effective solution, institutional consistency and ease of use for faculty while simplifying technical support. The pilot study participants include the UND College of Nursing and the Department of Music. Faculty and students will be surveyed at the close of the semester and the data will serve as a basis for discussion/improvement and provide direction for the future.

User Levels

The system supports four user levels: administrators, faculty, students, and guests. Student registration data for faculty is imported into the database at the beginning of the semester from the registrar. At import time, unique user ID’s and passwords are generated for each user (student). Faculty must attend a workshop for operational protocol and to receive their ID’s. Faculty can view student ID’s and passwords by using the*Show Students* function link in the left column of the screen. Administrators must be added manually, and guests are never added to the database. *Administrators* are allowed full editing access to all class sections.*Faculty* are allowed editing access to each class section they are assigned to teach.*Students* are allowed viewing access to all *active* areas of each class section
in which they are enrolled. Class section data areas are “*activated*” by faculty when data is first added. *Guests* are allowed viewing access to a subset of active areas for any class section.

Menus and Logins

A hierarchical menu system of colleges, departments, and courses provides access to each course section. Users login into a section with their assigned username and password through a login screen. On login, the user’s access level is determined, and is displayed on-screen at all times while active. Logins are maintained throughout a session until users either click the logout button, return to the menu system, or are timed out by failing to access course data for 120 minutes. If the user is identified as an instructor or administrator, they are given editing capability. In addition to the displayed data areas, input forms allow these users to edit, delete, or add to the data for the current section. Each edit or addition is time stamped, so students will know the currency of the data, and so an accurate record of submittal can be kept.

Content Areas
Seven data areas are currently implemented

- Syllabus - an area to post the course syllabus.
- Instructor Information - office, office hours, office phone, E-mail, and notes.
- Announcements - dated announcements to the class.
- Assignments - dated assignments with due dates.
- Discussion - a threaded discussion, where only faculty can post topic messages.
- Resources - an area to post old exams or articles as a study aid.
- Links - an area to save links to other web addresses.

Impact of Technology on Quantitatively Oriented Business Classes

JAYAVEL SOUNDERPANDIAN
Department of Business
University of Wisconsin - Parkside
Kenosha, WI 53406
Jayavel@usa.net

Several of the core business courses are quantitatively oriented. The more prominent one among them are: Business Statistics, Management Science, Operations Management, Managerial Accounting, Marketing Research and Investments. The tedium of the calculations involved in these courses has been a great hindrance to teaching and learning. Thanks to advances in computing technology, not only can the hindrances be eliminated, but also some impressive pedagogical innovations that enhance learning are possible. The tedium of calculation can be eliminated through the use of spreadsheets in general and spreadsheet templates in particular. These templates will be introduced and explained. A Solver macro for optimization problems in Management Science will also be explained. The use of special purpose software for simulation and scheduling techniques in Operations Management will be explained.

Impact of Technology on Quantitatively Oriented Business Classes

JAYAVEL SOUNDERPANDIAN
Department of Business
University of Wisconsin - Parkside
Kenosha, WI 53406
Jayavel@usa.net

Introduction

Several of the core business courses are quantitatively oriented. The more prominent ones among them are: Business Statistics, Management Science, Operations Management, Managerial Accounting, Marketing Research and Investments. The tedium of the calculations involved in these courses has been, in the past, a great hindrance to teaching and learning. Thanks to the advances that have taken place in computing technology, not only can the hindrances be eliminated, but also some impressive pedagogical innovations that enhance learning are possible. The purpose of this paper is to explain how, in most of the quantitatively oriented business classes, the tedium of calculation can be eliminated and the pedagogical innovations can be made. Although the explanations here are restricted to Business Statistics, Management Science and Operations Management classes, the ideas are easily entendable to other quantitatively oriented business courses as well.

The elimination of tedium is efficiently achieved through the use of spreadsheets in general and spreadsheet templates in particular. Accordingly, in what follows, several spreadsheet templates are introduced and explained. These templates are especially suited for the Business Statistics and related courses such as Marketing Research. A number of techniques in Operations Management can also be implemented on spreadsheet templates. Following the discussion on spreadsheet templates, the use of the Solver macro for optimization problems in Management Science classes is explained. Finally, the use of special purpose software for simulation and scheduling techniques covered in Operations Management classes is explained.

At the end of each discussion on the different techniques, the advantages and disadvantages of using the suggested method are listed. In the concluding section, a summary of how technology has affected both the content and delivery of business classes is given.

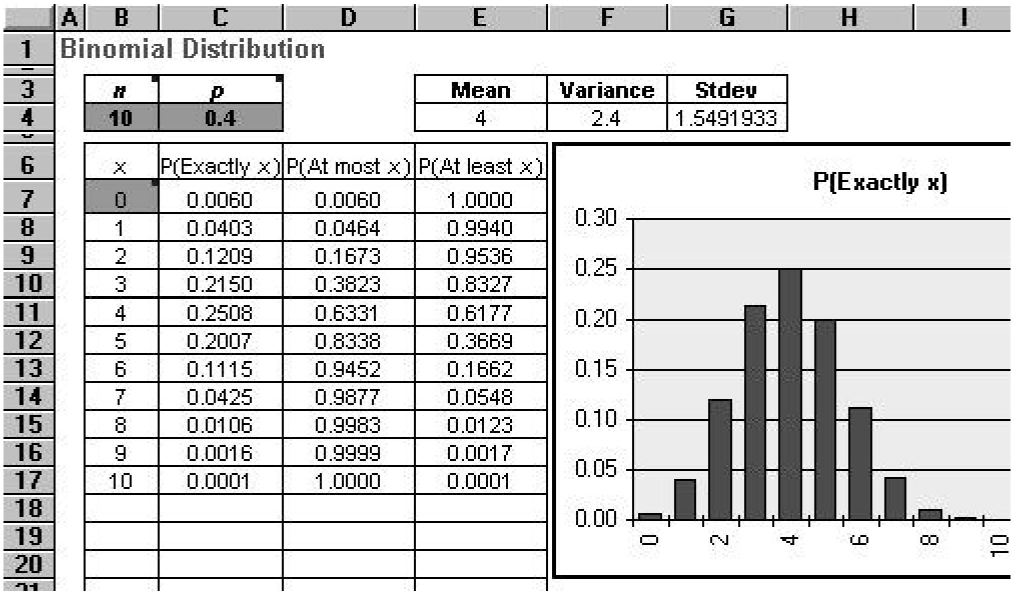
Spreadsheet Templates

A remarkable advantage of solving a problem on spreadsheets is that when the data of the problem is changed, the spreadsheet automatically updates to yield the revised solution. This leads naturally to the idea of spreadsheet templates, where all the cells except those containing the data are "locked". Only the cells containing the data are "unlocked" and therefore can be changed by the user. The formulas in the locked cells are kept intact. To solve a problem with a template, all that the user has to do is enter the data in the space provided for the data and read off the solution in another area. The tedium of calculation is completely eliminated. When I design templates, I shade the data areas in green so that the areas are clearly visible and the green color reminds the users that they can and should change only those cells. If there are instructions for the user, I leave them as cell comments (called cell notes in Excel version 5.0), or as texts in appropriate cells written in Magenta colored fonts. I have found that such color codings in templates are very effective.

The advantages of using templates stem not only from eliminating the tedium but also from the fact that spreadsheet commands such as Goal Seek and Data|Table can be used in conjunction with the templates. Indeed, the use of these commands make many kinds of meaningful and practical problems -- problems that were considered very hard in the past -- are easily solvable using spreadsheet templates.

As the first example of a spreadsheet template, the one I designed for Binomial distribution calculations is shown in Figure 1. The shaded cells contain data, which in this case are the number of trials *n* and the probability of success *p*. The comment/note indicators in cells B4 and C4 tell the user that there are some instructions for the user about what *n* and *p* are and how they are to be entered. Once they are entered, the table below the data shows all the probability results. For instance, when *n* = 10 and *p* = 0.4, the probability of at most 4 successes is 0.6331. The user is able to get these results without having to do any calculation and without having to know the spreadsheet formulas. The tedium vanishes.

The pedagogical innovations are easy to demonstrate now. To show the effect of increasing *n* or decreasing *p*, all that the instructor needs to do is make that change in cell B4 or C4 and show how the probabilities changed. To make facilitate this type of sensitivity analysis, the bar chart of probabilities has been added to the template.



Binomial Distribution

n	*p*
10	0.4

Mean	Variance	Stdev
4	2.4	1.5491933

x	P(Exactly x)	P(At most x)	P(At least x)
0	0.0060	0.0060	1.0000
1	0.0403	0.0464	0.9940
2	0.1209	0.1673	0.9536
3	0.2150	0.3823	0.8327
4	0.2508	0.6331	0.6177
5	0.2007	0.8338	0.3669
6	0.1115	0.9452	0.1662
7	0.0425	0.9877	0.0548
8	0.0106	0.9983	0.0123
9	0.0016	0.9999	0.0017
10	0.0001	1.0000	0.0001

Figure 1. The Binomial Distribution Template

The template is in many ways superior to the Binomial probability tables typically found in statistics textbooks. First, the template will work accurately for *n* values up to 500. Second, the *p* value can be any value between 0 and 1. Third, and this is where it gets interesting, the template can be used in conjunction with Goal Seek command to answer some very practical questions. Most textbook questions typically would simply ask for some Binomial probability and the question ends there. Whether the result is too small or too large in practice, and how the probability can be influenced by changing *n* or *p* is never questioned because such questions call for tedious calculations. Since I use the above template in my classes I am able to ask questions such as those seen in the problem below. This problem is taken from [Sounderpandian 1996].

> A salesperson goes door-to-door in a residential area and demonstrates the use of a new household appliance. At the end of a demonstration, there is a constant 0.2107 probability that the potential customer would place an order for the product. To perform satisfactorily on the job, the salesperson needs at least 4 orders. Assume that each demonstration is a Bernoulli trial.

a. If the salesperson makes 15 demonstrations, what is the probability that there would be exactly 4 orders?

b. If the salesperson makes 16 demonstrations, what is the probability that there would be at most 4 orders?

c. If the salesperson makes 17 demonstrations, what is the probability that there would be at least 4 orders?

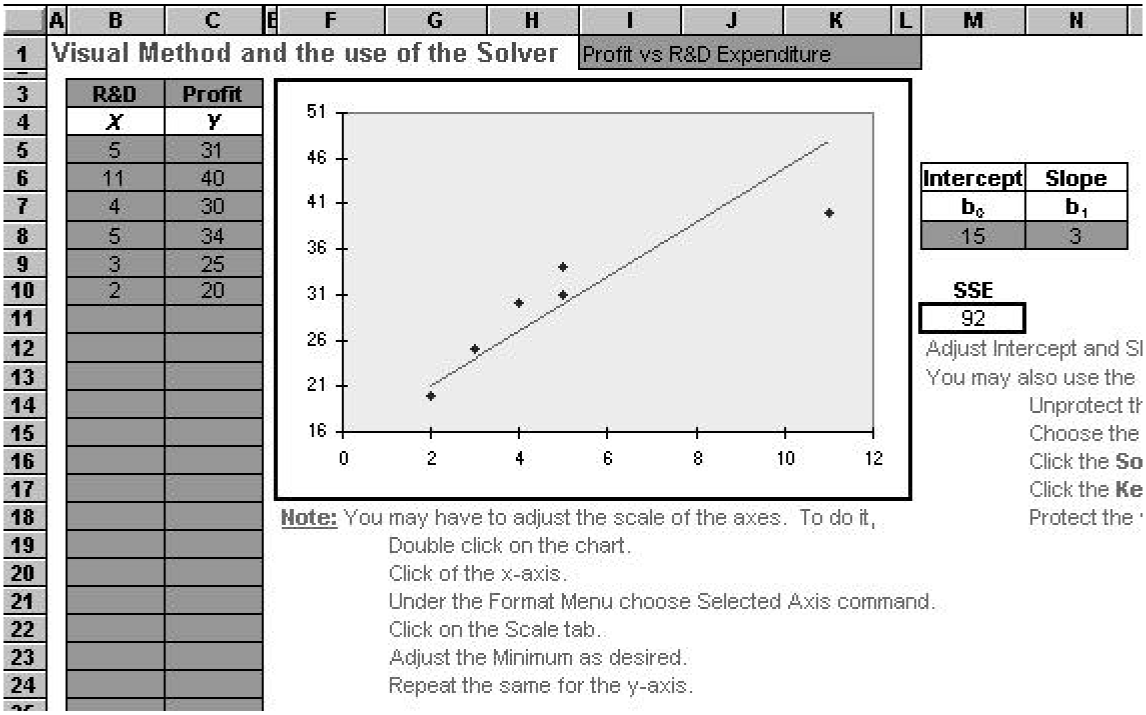
d. If the salesperson makes 18 demonstrations, what is the probability that there would be anywhere from 4 to 8 (both inclusive) orders?

e. If the salesperson wants to be at least 90% confident of getting at least 4 orders, at least how many demonstrations should she make?

f. The salesperson has time to make only 22 demonstrations, and still wants to be at least 90% confident of getting at least 4 orders. She intends to gain this confidence by improving the quality of her demonstration and thereby improving the chances of getting an order at the end of a demonstration (currently this probability is 0.2107). At least to what value should this probability be increased in order to gain the desired confidence? Your answer should be accurate to 4 decimal places.

Another good example of pedagogical innovation is seen in Figure 2 which shows the template for carrying out a Simple Regression using a "visual" approach, taken from [Sounderpandian 1998a]. In this template, for the *X* and *Y* data entered in columns B and C, the regression line needs to be found. To explain the concept of minimizing the sum of squared errors (SSE), the instructor can ask the students to "play with" the intercept and the slope of the regression line in cells M8 and N8 until the SSE in cell M11 is minimized. On doing it, the students realize the nuances of the problem. In particular, they realize that if the intercept is optimized for a given slope, and then the slope is changed, the intercept is no longer optimal. Thus the problem requires jockeying back and forth between intercept and slope and thus is a hard problem. At the same time, they are also able to see in the accompanying chart how well the current regression line fits the scatter of points. I have found this exercise to be a very useful lesson. The lesson would be impossible without the use of spreadsheet templates.

I complete the discussion on Simple Regression with the use of more templates and carefully designed exercises and projects. The use of Solver is discussed in a later section.



Visual Method and the use of the Solver

Profit vs R&D Expenditure

R&D	Profit
X	*Y*
5	31
11	40
4	30
5	34
3	25
2	20

Intercept	Slope
b_0	b_1
15	3

SSE

92

Adjust Intercept and Sl

You may also use the

Unprotect th

Choose the

Click the **So**

Click the **Ke**

Protect the

Note: You may have to adjust the scale of the axes. To do it,

Double click on the chart.

Click of the x-axis.

Under the Format Menu choose Selected Axis command.

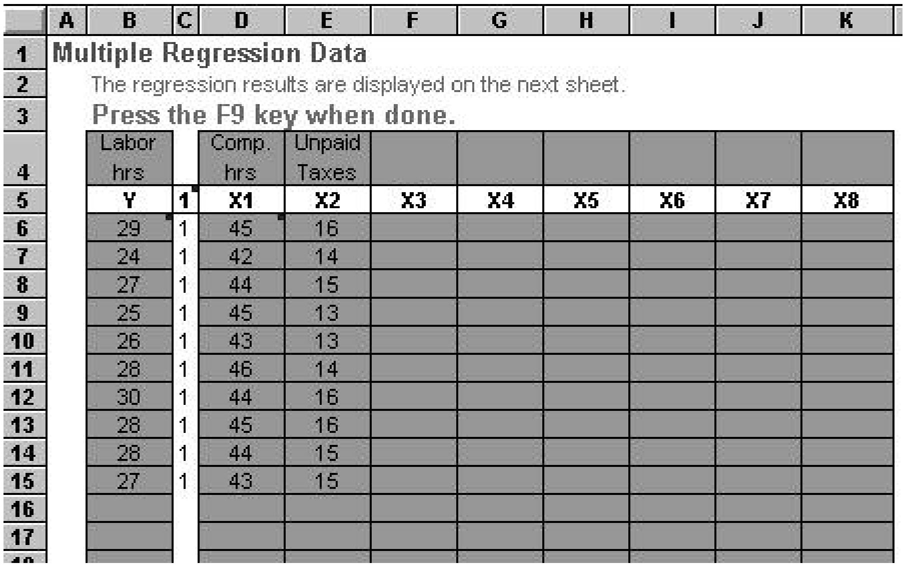
Click on the Scale tab.

Adjust the Minimum as desired.

Repeat the same for the y-axis.

Figure 2. Simple Regression by Visual Method

A final example in this section is the template I designed for Multiple Regression. This template consists of a Data sheet, shown in Figure 3, and a Results sheet, shown in Figure 4. The user enters the data in the shaded area which can accommodate up to eight independent variables, *X1* to *X8*, and up to 100 observations. As the instruction reads, the user presses the F9 key after entering all the data. The results are then calculated and displayed on the Results sheet seen in Figure 4. The results contain the regression coefficients, their standard errors, their *t* ratios and the *p*-value associated with those *t* ratios. In addition, there is a Forecast area, where for given values of the independent variables a confidence interval is calculated and displayed. Finally, there is an ANOVA table (not seen in the figure), numerical values of the residuals (not seen in the figure) and a plot of the residuals. A little thought would reveal that this way of conducting a multiple regression experiment is superior in many ways to conducting it using special purpose software such as SPSS or SAS. Here the user does not have to use any special commands called for in SPSS or SAS. The user can change the data and see its effect immediately.



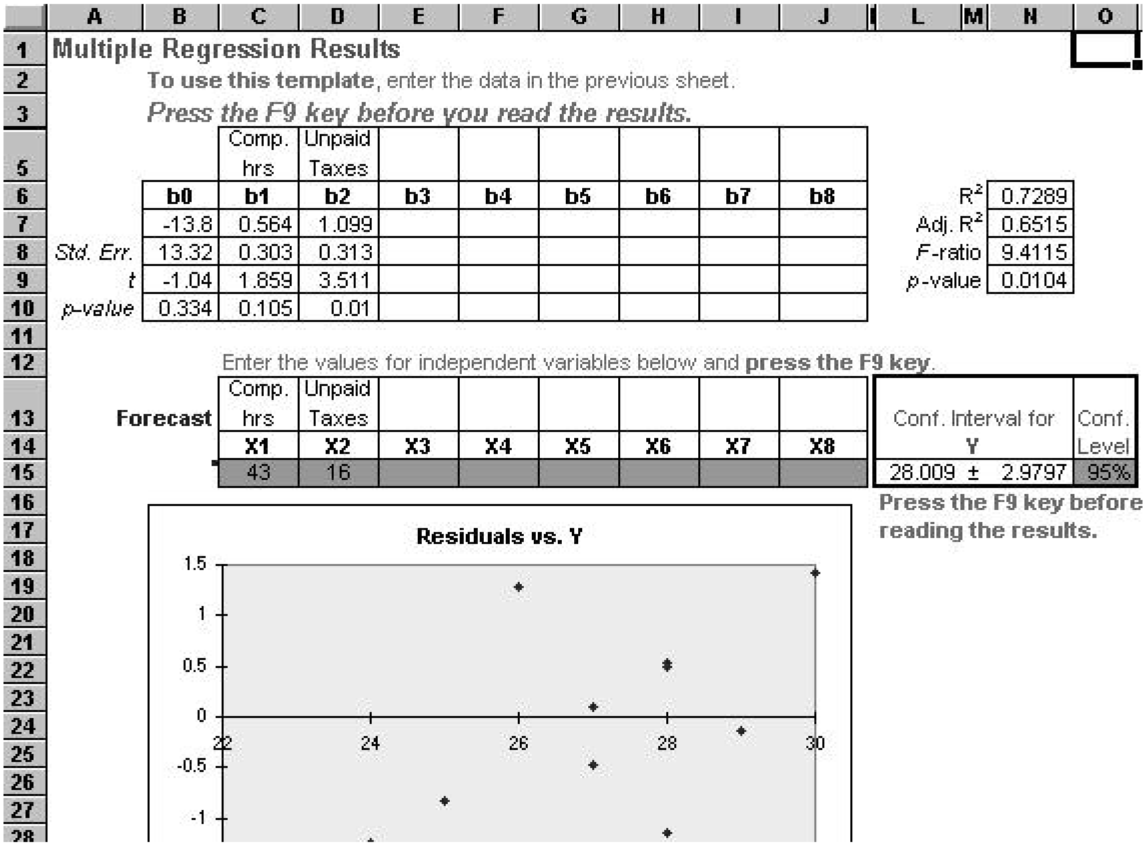
	A	B	C	D	E	F	G	H	I	J	K
1	Multiple Regression Data										
2		The regression results are displayed on the next sheet.									
3		Press the F9 key when done.									
4		Labor hrs		Comp. hrs	Unpaid Taxes						
5		Y	1	X1	X2	X3	X4	X5	X6	X7	X8
6		29	1	45	16						
7		24	1	42	14						
8		27	1	44	15						
9		25	1	45	13						
10		26	1	43	13						
11		28	1	46	14						
12		30	1	44	16						
13		28	1	45	16						
14		28	1	44	15						
15		27	1	43	15						
16											
17											

Figure 3. The Data Sheet of Multiple Regression Template

Here also some innovations are possible. For instance, one independent variable can be a function of another independent variable. For instance, X3 may be equal to $X2^2$, thus leading to polynomial regressions. When one independent variable is a linear composite of a few others, we have the classic case of multicollinearity. I use this fact to illustrate the serious consequences of multicollinearity by making an independent variable exactly or almost exactly equal to a linear composite of one or two other independent variables. Such a demonstration is difficult or impossible without the use of spreadsheets.

The plot of residuals is also useful in explaining the phenomenon of heteroscedasticity. By taking a "normal" multiple regression example and then transforming the *Y* variable into its reciprocal, I have been able to demonstrate the effect of heteroscedasticity. Multicollinearity and heteroscedasticity are two difficult concepts to explain while teaching Multiple Regression. The use of spreadsheet templates makes them much easier.

I have shown the innovations with two examples. But in almost every one of the spreadsheet templates I use in Business Statistics and Operations Management classes, I have always been able to find some pedagogical advantages of using spreadsheet templates. The reader can easily find similar possibilities in other templates as well.



Multiple Regression Results

To use this template, enter the data in the previous sheet.

Press the F9 key before you read the results.

	Comp. hrs	Unpaid Taxes						
b0	**b1**	**b2**	**b3**	**b4**	**b5**	**b6**	**b7**	**b8**
-13.8	0.564	1.099						
Std. Err. 13.32	0.303	0.313						
t -1.04	1.859	3.511						
p-value 0.334	0.105	0.01						

R²	0.7289
Adj. R²	0.6515
F-ratio	9.4115
p-value	0.0104

Enter the values for independent variables below and **press the F9 key**.

	Comp. hrs	Unpaid Taxes						
Forecast	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**X8**
	43	16						

Conf. Interval for Y	Conf. Level
28.009 ± 2.9797	95%

Press the F9 key before reading the results.

Figure 4. The Results Sheet of Multiple Regression Template

Advantages and Disadvantages of Spreadsheet Templates

As seen from the above examples and explanations, the advantages of using spreadsheet templates for teaching quantitatively oriented topics are:

1. The spreadsheet software is ubiquitous and the user will not be at a loss when he or she moves to a new place. Special purpose software are not ubiquitous.

2. At times, the spreadsheet is better than even a special purpose software since the user can modify the data easily and immediately see the effect on results.

3. It is possible to explain difficult concepts with proper use of charts and tables in a spreadsheet template.

4. A spreadsheet template can be used in conjunction with such commands as Goal Seek and Data|Table. With their use, it is possible to answer practical questions, which would be very hard or impossible without the use of spreadsheets.

There are some disadvantages as well. They are:

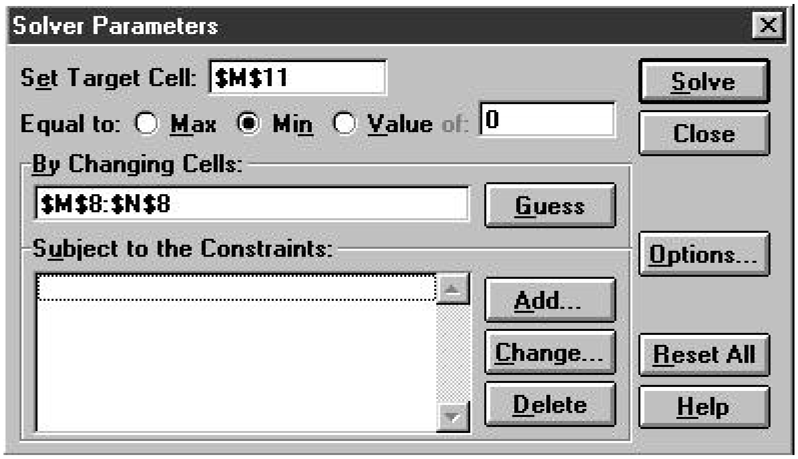
1. The greatest disadvantage with spreadsheet templates is that the students may treat them as black boxes. They might mechanically carry out a calculation without understanding the underlying concepts. At times, they may misuse it by applying a technique to a case which is not amenable to the technique.

2. The second disadvantage is scalability. In the Multiple Regression example, for instance, the template can accommodate only eight independent variables and 100 observations. If the data exceeds any of these limits the user may not know what to do. It is true that a user well-versed with spreadsheets can extend the data size by making proper modifications to the template. But most of the time, the user does not know how to do it.

Using the Solver Macro in Spreadsheets

The use of Solver can be explained with the Simple Regression template seen in Figure 2. In that example, the user needs to find the slope and the intercept of the regression line that would minimize the SSE. Doing it manually is tiresome. At this point the instructor can introduce the Solver macro available in Excel and other spreadsheet software. The Solver can be invoked to carry out a systematic search for the best values of the slope and the intercept. Figure 5 shows the settings on the Solver dialog box to accomplish this. The cell references in this dialog box refer to the spreadsheet in Figure 2.

Doing this exercise immediately after asking the students to manually search for the best values enhances the learning. Students understand appreciate the power of the Solver and the convenience of using it to solve optimization problems.



Solver Parameters

Set Target Cell: $M$11

Equal to: ○ Max ● Min ○ Value of: 0

By Changing Cells: $M$8:$N$8

Subject to the Constraints:

Figure 5. The Solver Dialog Box Entries for the Regression Problem

In Management Science classes, several of the topics require optimization. Typical problems would require something good such as profit or revenue to be maximized or something bad such as cost or tax to be minimized. In all those instances, as long as the profit or the cost is a smooth function of the changing variables, the Solver can be efficiently used. The textbook [Ragsdale 1998] contains details of using the Solver in this manner.

Besides the Solver, an instructor may also use spreadsheet add-ons such as What's Best and/or Insight.xla. These add-ons complement the Solver so that the user can handle problems that are "discrete", such as those that involve networks. The use of such add-ons has revolutionized teaching Management Science classes. Today, the students routinely solve problems that were considered too hard to solve in classroom only a few years ago.

Advantages and Disadvantages of Using the Solver

The advantage of using the Solver is that the instructor can teach more and teach better, because the Solver makes optimization problems easy for students to handle. The disadvantage of using the Solver, once again, is that it could be a black box to the students. The instructor should explain to the students the fundamentals of optimization and how exactly the Solver itself works. Also, the limitations such as the limit on number of variables and the number of constraints should be made clear. The cases of unbounded and infeasible solutions should be covered. In the case of nonlinear optimization, the possibility of local optimum as opposed to global optimum should be pointed out.

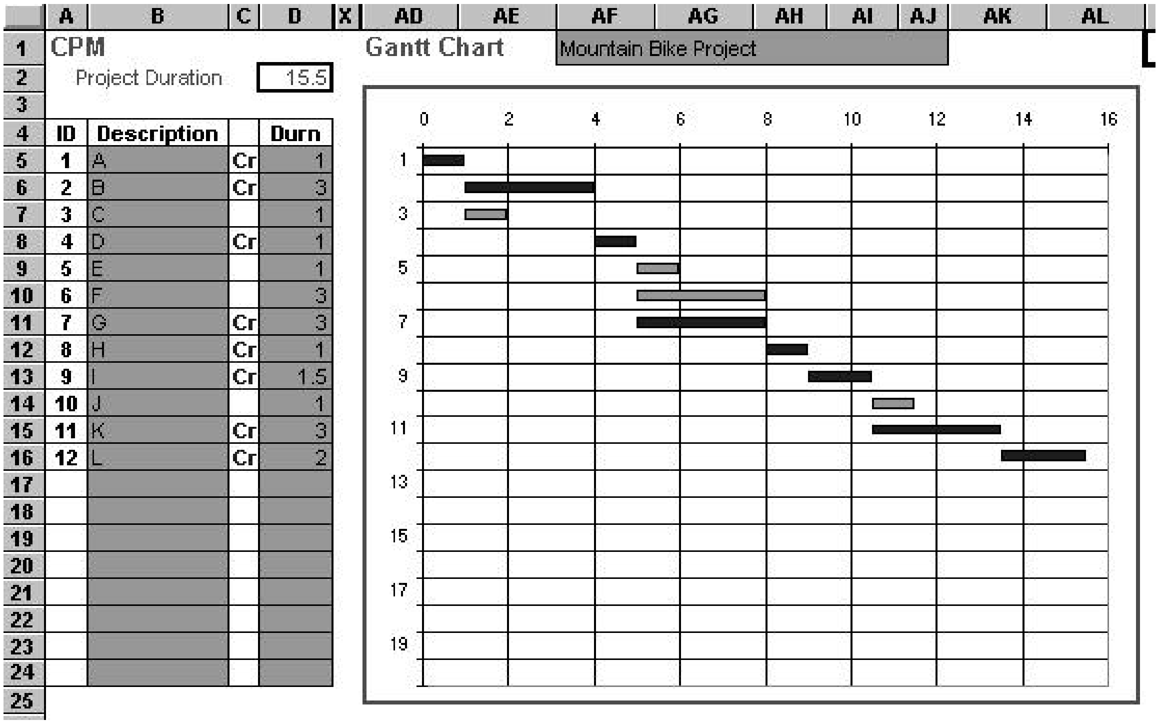
Spreadsheet Templates for Operations Management

Another business course that involves several quantitative techniques is Operations Management. In [Sounderpandian 1998b] I have shown several templates that can be used in this course. Two of them are detailed below.

Figure 6 shows the spreadsheet template for project scheduling. The user enters the list of tasks in the project in column B and their durations in column D. The user also enters the precedence relationships such as task X should precede task Y in columns F through V (not shown in the figure). Once these data are entered the template automatically schedules the project and displays a Gantt chart of the optimal schedule. Thus the tedium of scheduling is removed and the student is free to explore other managerial questions such as which tasks are worth crashing so as to reduce the project duration.

A topic that many textbooks omit is the updating of the schedule as the project progresses, because it involves tedious explanations. With the use of the templates, updating is quite easy. Completed tasks can be easily removed, or their durations can be set to zero. Partially completed tasks will have their durations updated to reflect only the time needed to complete it. Once again, the use of templates enables the instructor to teach more and teach better.

The instructor can overcome the danger of students treating the template as a black box by explaining the concepts of scheduling before introducing the template. Examinations administered in a course should include questions that test the understanding of concepts and not just the use of the template.



CPM

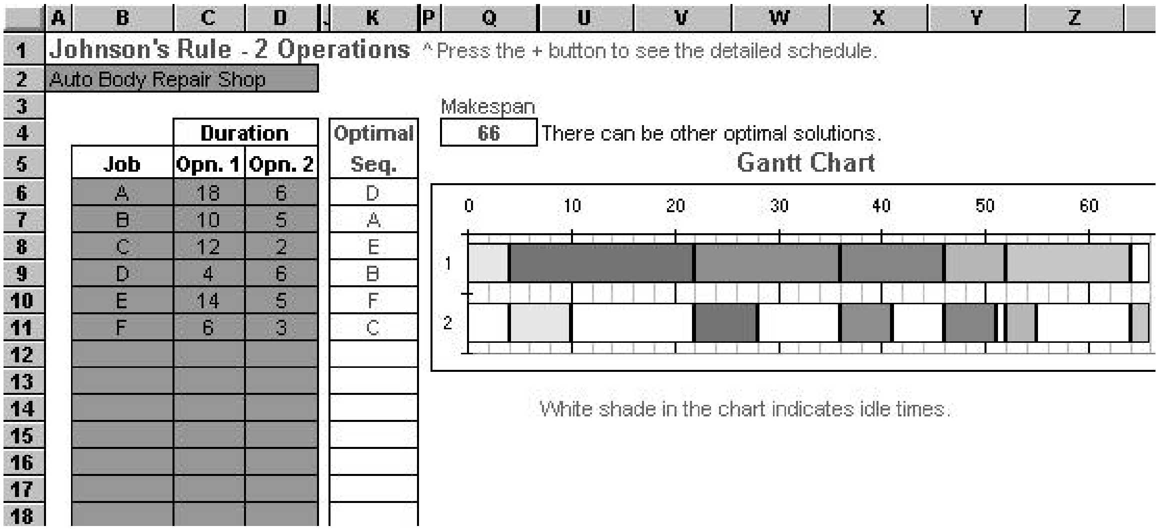
Project Duration 15.5

ID	Description		Durn
1	A	Cr	1
2	B	Cr	3
3	C		1
4	D	Cr	1
5	E		1
6	F		3
7	G	Cr	3
8	H	Cr	1
9	I	Cr	1.5
10	J		1
11	K	Cr	3
12	L	Cr	2

Figure 6. A Spreadsheet Template for Project Scheduling

An interesting point to note about the above template is that although it implements a scheduling algorithm, no programming such as a Visual Basic (VB) macro has been used in it. All calculations are carried out by ordinary Excel formulas in cells. This makes it easy for a student to understand the design of the template just by examining the formulas in cells.

The next example is a template designed to implement Johnson's algorithm for scheduling *n* jobs through 2 machines. No VB macro has been used in this template, so that a student can learn the calculations simply by examining the formulas in cells.



Johnson's Rule - 2 Operations ^ Press the + button to see the detailed schedule.

Auto Body Repair Shop

Makespan 66 There can be other optimal solutions.

	Duration		Optimal
Job	Opn. 1	Opn. 2	Seq.
A	18	6	D
B	10	5	A
C	12	2	E
D	4	6	B
E	14	5	F
F	6	3	C

White shade in the chart indicates idle times.

Figure 7. The Template for Johnson's Rule

To use this template the user enters the data in the columns B, C and D. The optimal sequence of jobs appears in column K and a bar chart showing the loading of each job in each machine appears on the right. The “makespan”, or the time needed to complete all the jobs appears in cell Q4.

The template is useful in demonstrating the sensitivity of the makespan to the different task durations. For instance, if the time required for any *one* of the jobs in Machine 1 can be reduced by 2 units which one should be reduced? Without the template, it is a very hard question. With it, it is a simple trial and error method.

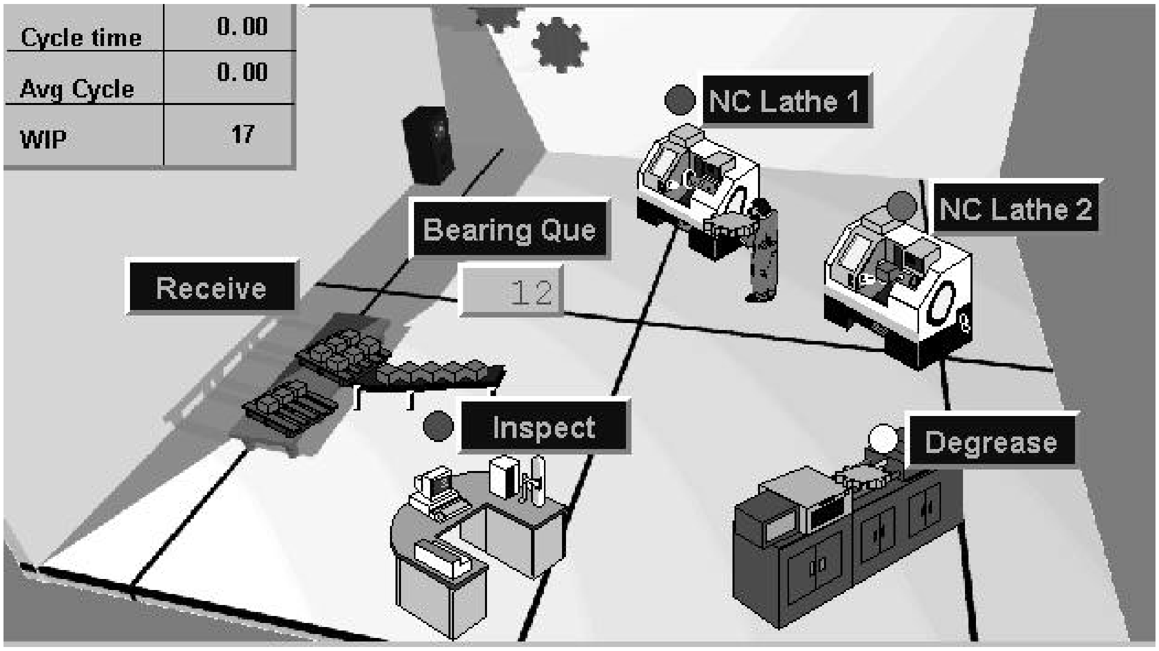
The Role of Special Purpose Software

So far only the spreadsheet templates have been discussed. Impressive advances have also taken place in special purpose software related to business courses. As an example, ProModel for simulation is discussed below.

ProModel

ProModel is a powerful and user-friendly software that can be used to model manufacturing, service, transportation and warehousing operations. In the past, the software available for simulation were general purpose ones such as FORTRAN or BASIC which required mastery of computer programming. More recently, the so called special purpose software such as GPSS or SLAM appeared. But even these required the user to be experienced in computer programming. ProModel, on the other hand, requires no computer programming expertise at all. Using menus, icons and graphics, the user can easily create a simulation model, complete with all animations.

Figure 8 shows a ProModel simulation of a small job shop where jobs arrive and are kept in a the storage area. They are processed through either one of the two lathes, degreased and inspected. When the model is run the user sees the movement of parts and the worker through a complete animation. At the end of the simulation, statistics such as the utilization of the machines and the worker are printed out.



Cycle time	0.00
Avg Cycle	0.00
WIP	17

Figure 8. The Layout of a ProModel Simulation

In real life, the time taken to complete a task, the occurrence of machine breakdowns and the occurrence of a defective part are all unpredictable. These events are simulated in the model through the use of random numbers. The model can be used to see what happens to capacity and utilization figures if an additional lathe or an additional operator is added. Thus the model can be very useful for managerial decision making.

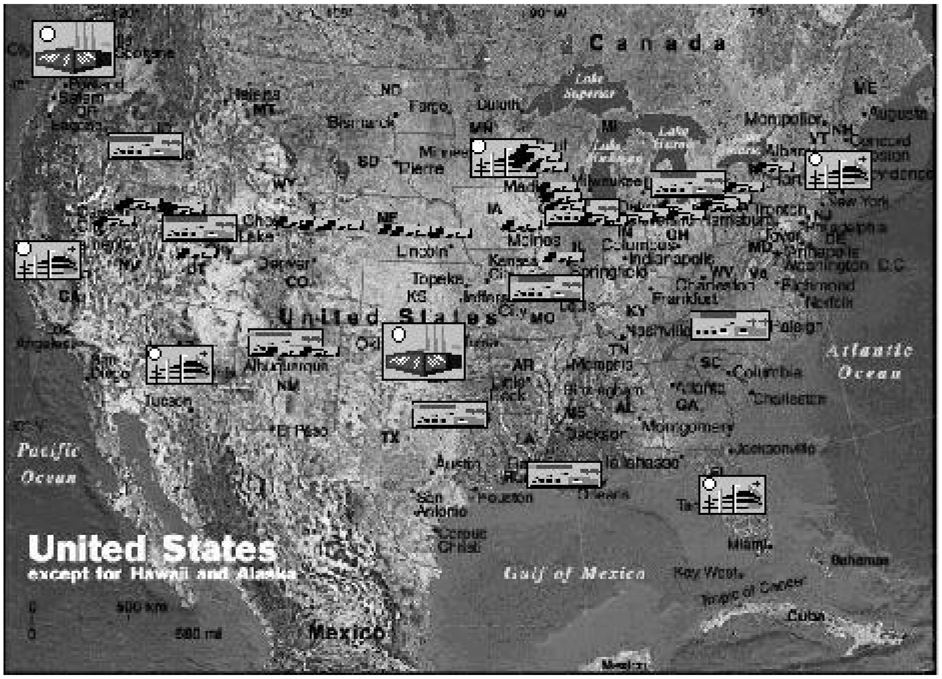


Figure 9. The Layout of another ProModel Simulation

Figure 9 shows the ProModel screen for a distribution model. When this model is run the one sees the movement of several trucks geographically distributed across the nation. By simulating the operation of the trucks under different scheduling policies, it is possible to find the best policy that maximizes utilization of the trucks and the service levels of the depots served. The simple observation of the animation on the screen can help the user to identify inefficiencies and potential opportunities for improvement.

Advantages and Disadvantages of Using Special Purpose Software

The advantage of special purpose software is that they can remove the tedium of computer programming. Thus difficult computational techniques, such as simulation, become possible for a non-programmer.

The disadvantage is non-portability. The software is likely to be not available everywhere. ProModel, although popular, has several competing software such as SIMSCRIPT. Thus, a student who learned ProModel may not be able to use it in a workplace which does not have ProModel.

Impact on Delivery

The most important impact that technology has had on the delivery of courses is Distance Education. Through the use of compressed TV and the Internet, it is now possible to deliver a course to student at remote locations. Supporting software such as LearningSpace from the Lotus Institute help to place a course on the Internet.

The University of Wisconsin - Parkside, through a reciprocal arrangement with sister campuses at La Crosse and Eau Claire, offers a few MBA courses every semester as Distance Education courses to students at the other two campuses. This increases the frequency of offerings and reduces the time needed by the students to graduate.

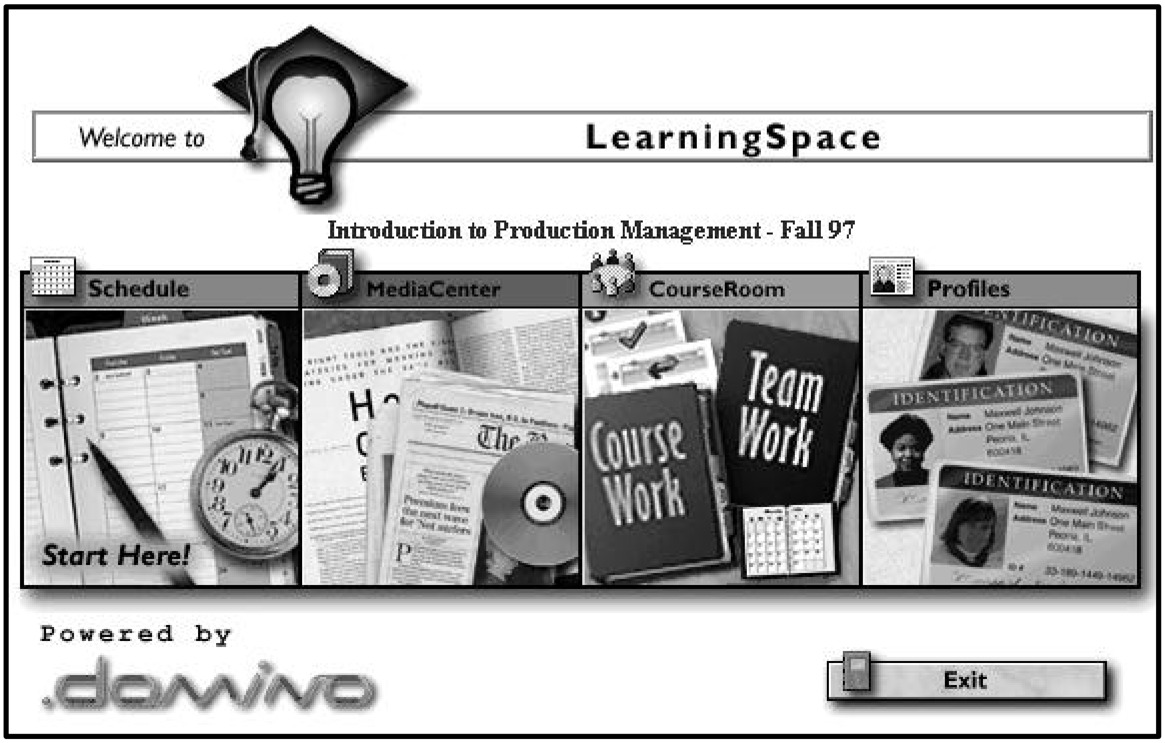


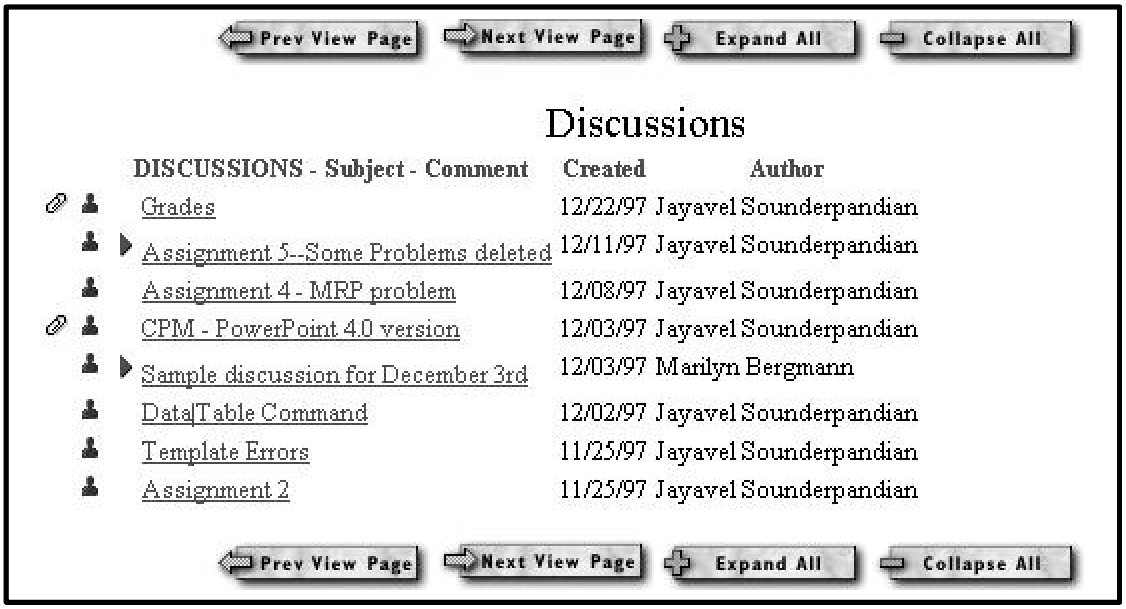
Figure 10. The LearningSpace Screen

Figure 10 shows the initial screen of LearningSpace for a particular course. The instructor can set up the Schedule area with details on class meetings, the MediaCenter with reading as well as audio and video materials, the CourseRoom for on line discussions and the Profiles area for general information about the students in the class. Because the access to LearningSpace is password protected, an instructor can put copyrighted materials in the MediaCenter.

On clicking the CourseRoom button in the above screen, the user will see the CourseRoom come into view. Figure 11 shows the CourseRoom, the most volatile area where on line discussions can be carried out. Each line on the listing can be clicked to see the discussion along that thread. A student can add his/her own comments, delete his/her own comments in this area. It is also possible for students to attach to their entries any files containing documents or spreadsheets or other matters. The paper clip icons to the left of some entries in the figure denote that there are attachments in that discussion.

The “twisties” to the left of some items, e.g., Assignment 5, indicate that there are additions to the initial post, and thus there indeed is a discussion going on about that topic.

There are now several competing software that can also be used for placing a course on the web. In my opinion, LearningSpace is the most user-friendly and the costliest. As always, these characteristics can change very quickly.



Discussions

DISCUSSIONS - Subject - Comment	Created	Author
Grades	12/22/97	Jayavel Sounderpandian
Assignment 5--Some Problems deleted	12/11/97	Jayavel Sounderpandian
Assignment 4 - MRP problem	12/08/97	Jayavel Sounderpandian
CPM - PowerPoint 4.0 version	12/03/97	Jayavel Sounderpandian
Sample discussion for December 3rd	12/03/97	Marilyn Bergmann
Data|Table Command	12/02/97	Jayavel Sounderpandian
Template Errors	11/25/97	Jayavel Sounderpandian
Assignment 2	11/25/97	Jayavel Sounderpandian

Figure 11. The CourseRoom in LearningSpace

Concluding Remarks

The advances in technology have greatly affected the content and delivery of Business Statistics, Management Science and Operations Management courses in business schools. The changes have been brought about by advances in spreadsheet software, special purpose software and computer hardware. The advances that have taken place in compressed video technology and the Internet have triggered a rapid growth in the number of courses offered as Distance Education courses. To keep up with these developments, instructors need to keep in constant touch with technology and adapt quickly to those technologies.

References

[Ragsdale 1998] Ragsdale, Cliff T. 1998. Spreadsheet Modeling and Decision Analysis, Southwestern College Publishing, Cincinnati, OH.

[Sounderpandian 1996] Sounderpandian, Jayavel. 1996. StatSheets, Irwin, Chicago, IL.

[Sounderpandian 1998a] Sounderpandian, Jayavel. 1998. Excel Supplement to Statistics for Management by Levin and Rubin, Prentice Hall, Upper Saddle River, NJ.

[Sounderpandian 1998b] Sounderpandian, Jayavel. 1998. Excel Supplement to Operations Management by Markland et al., Southwestern College Publishing, Cincinnati, OH.

Multi-facet Distance Education Services for Educational Leadership

31st Annual Conference in Fargo, ND and Moorhead, MN (April 16 - 18, 1998)
Small College Computing Symposium (SCCS98)
"Building the Virtual Campus: Measuring Our Progress"

TRI-COLLEGE UNIVERSITY EDUCATIONAL LEADERSHIP PROGRAM FACULTY:

Ronald M. Stammen, North Dakota State University,
Dennis W. Van Berkum, Moorhead State University ,
and Burton N. Nygren, North Dakota State University

Abstract

The purpose of the panel discussion was to share information and answer questions about how an educational leadership program has evolved during the past seven years toward providing multi-faceted distance education opportunities. The faculty is involved in and/or responsive to North Dakota State Universitys' Information Technology Roundtable which developed proposals to restructure university-wide services to accommodate and promote distance education. Likewise, faculty have utilized email extensively for groupwork. This year one member delivered a distance education inservice courses about technology in the classroom via Public Television statewide microwave towers. The faculty has begun to utilize webpages for instructional purposes to supplement their regular classroom and Interactive Video (Television) delivery The Tri-College University initiated an on-line web access for the entire program to enhance communications for recruitment and service to graduate students currently in the program. See http://www.ndsu.nodak.edu/ndsu/tricollege/

Multi-facet Distance Education Services for Educational Leadership

31st Annual Conference in Fargo, ND and Moorhead, MN (April 16 - 18, 1998)
Small College Computing Symposium (SCCS98)
"*Building the Virtual Campus: Measuring Our Progress*"

TRI-COLLEGE UNIVERSITY EDUCATIONAL LEADERSHIP PROGRAM FACULTY:

Ronald M. Stammen, North Dakota State University,
Dennis W. Van Berkum, Moorhead State University ,
and Burton N. Nygren, North Dakota State University

Introduction

The Tri-College Educational Leadership program has been involved in distance learning for the past seven years. The unique nature of the consortium, in which this program resides, provided several opportunities to facilitate the growth of distance teaching and learning. A specialist program has been offered over the North Dakota Interactive Video System (IVN). A cooperative venture with the University of North Dakota has evolved to offer a masters' degree program to residents in North Dakota. In Minnesota, both the masters' and specialist degree program are offered over a regional interactive television network.

The insights gained during the past seven years by faculty have been many. Success and failure have resulted in myriad of suggestions for others from an organizational stand point. The importance of proper planning, development, implementation, and assessment cannot be underestimated. These components combined with the anticipation of continued improvement make for an exciting environment. New approaches to course development and instruction, program scheduling, and marketing have evolved. The following are courses were developed for distance learning:

Educational Leadership Courses Developed by both Universities for Distance Education

Law and Organizational Structure

Curriculum, Instruction & Learning Theory

Technology and Information Systems

Personnel, Supervision, & Staff Development

Personal Communications & Ethics

Policy & Educational Finance

Research Measurement & Evaluation

Social, Culture & Political Communication

Scheduling is now a collaborative effort where schedules are created well into the future (Four years). Courses are aligned with participating institutions. Interactive system availability is considered prior to publishing of scheduling. New markets and approaches to marketing are being developed. Markets are now viewed relative to potential students numbers, access of appropriate technology, and the ability to maintain personal contact. Currently three different market areas are identified and served.

A management process for developing multimedia modules

The USWEST project was entitled,“Collaboratively Creating Multimedia Modules for Teachers and Professors. ” This curriculum development process involved teams of experts in teaching, computer science, and multimedia technology. Primary focus of the project was to teach college professors and K-12 teachers how to utilize multimedia to enhance their classroom instruction. [http://www.ndsu.nodak.edu/instruct/stammen/uswest]

This project was a challenge for faculty and administrators who were not accustomed to developing their own text material for students. Most educators are not trained to develop their courseware. Many members had experience preparing content in sequential, print-based media but most lacked expertise to technologically place such data for non-sequential, electronically dynamic interaction. The instructional requirements for each learning task contain specific performance standards which are utilized to determine training approaches for developing learning objectives, performance measures, and aspects of the training program regarding faculty needs, needed equipment, and subsequent support.

This systematic procedure establishes the sequence for developing competency profiles, related materials, supportive media, modules and/or learning guides, and field-tested procedures to judge the extent competency-based (learning) objectives are achieved in the process. This is done according to an array of outcomes and specific recommendations which are part of the Program Evaluation and Improvement Plan developed for both the pilot and actual implementation phases. The SCID procedure used a DACUM process (see inset) to outline a series of duties. A task analysis procedure determined how many tasks it would take to accomplish a duty. Two people were assigned to each duty to create a module.

In summary, the SCID process captured the content before the multimedia development strategy commenced to create electronic modules. Evaluation procedures were used throughout the project. Team members responding to these inquiries indicated that the SCID process proved to be an ideal management process among part-time people who worked on an irregular timetable. It kept all on task, but most important, bonded everyone to a common language with a specific purpose.

The results from field testing the thematic-based pilot projects utilizing customized multimedia carts in five K-12 and two higher education institutions were compiled during the summer of 1997. Bound copies are limited and available at FLC 210, North Dakota State University. This website and data are used for Ed 722 Instructional Media and Materials and Ed 733 Technology and Networking

Five Phases of Systematic Curriculum and Instructional Development (SCID) used to Manage the Project:

Phase I: Analysis - uses needs analysis, job analysis, and task verification processes. (Utilizes a special process called DACUM (an acronym meaning Develop A CUrriculuM)

Phase II: Design - outlines the overall curriculum and develops the foundation for the training program.

Phase III: Instructional Development - determines what will be taught and what learning activities, materials, and instructional methods will be used; develops the modules which are comprised of learning guides including supportive multimedia methods; field testing; pilot testing and revision processes.

Phase IV: Training Implementation - activates the training plan, its evaluation, and documents learner achievement.

Phase V: Program Evaluation - evaluates each of the five phases, including product, phase, and process evaluation.

(The Systematic Curriculum Instructional Development (SCID) model (©1990) was created at the Center for Education and Training for Employment (CETE) at The Ohio State University, Columbus, Ohio.)

Delivering Educational Leadership Programs via Distance

Two-Way Interactive Video

The TCU Educational Leadership (AKA School Administration) program went on-line late fall of 1990 when Ron Stammen taught School Plant Planning and Maintenance (12 students). The North Dakota Interactive Video Network (IVN) made its debut that fall with T-1 transmitted voice-video transmission across the state. The first three-cycle for the specialist program ended Spring 1993. Although class sizes ranged from 10 to 40, it was not until the second three-cycle (1993-1996) when students begin obtaining this post-masters degree by attending most courses via IVN. Jim Blomberg, an administrator at Minot became the first who acquired his specialist degree via IVN timetable. Most other graduate students expedited their degree through the accelerated summe r session which has two one-week sessions and two three-week sessions from the first week in June to the first week in August. The cooperative masters degree program with the University of North Dakota had its first two-year cycle during 1996-1997 and 1997-1998. A second two year cooperative for central and western North Dakota begins this fall.

Denny Van Berkum started the ITV program in west-central Minnesota locations during the 1996-1997 school year. This ITV program became a regular part of the rotating schedule starting with the fall of the 1997-1998 term.

Computer-mediated Instruction

SENDIT, a statewide K-12 e-mail host service was initiated by the TCU faculty through grants from the North Dakota Educational Telecommunications Council. The funding for this project commenced in 1991 and continued until 1995 when Ron Stammen, on behalf of the Council and the University draft a memorandum of understanding which transferred the operation over to NDSU Information Technology Services via a line-item on the budget of the Department of Public Instruction. This medium, SENDIT, has been utilized extensively since 1992 to provide communication between advisor and advisee for answering questions, developing program-of-study, and

Broadcast Television

Prairie Public Television facilitated an online course from their national Public Broadcasting resources utilizing satellite delivery. The course entitled “Multimedia in the Classroom” with Ron Stammen as the instructor of record for graduate credit. One hundred and eighty-six educators from North Dakota and Minnesota took the course by downloading the course presentations to their video tape recorders during one night each week.

Course-in-a-box an d Video

The Master Teacher Corporation in conjunction with NDSU Continuing Education Division offers a series of courses for graduate in-service credit nationwide. Ron Stammen is the instructor for the course entitled, “Strategies for Building Home-School Relationships.” Sections from this course is also utilized for a portion of the masters degree course

Concerns regarding the implementation of Distance Education Programs

Faculty members contemplating having their programs delivered via distance education technologies should seriously contemplate whether they have the capacity and time to handle the following in addition to developing a marketing plan:

Marketing

Instructional Development

Promoting

Developing Curricula for the Media Selected

Planning Logistics

Scheduling timetables

Recruiting

Outreach Meetings

Coordination of Assessment

Meetings for Preparation

Procedures for Testing

The Pain and Promise of Electronic Educational Administration Programs

The Great Plains Deans' Consortium sponsored a "Faculty Development LearnShop" on DISTANCE EDUCATION in Lincoln, Nebraska (May 12-16,1997). The primary focus was putting courses on the Internet. The Great Plains Deans' Consortium is a regional consortium of land grant institutions that provides graduate and undergraduate education through extended education programs. The institutions involved are Iowa State University, Kansas State University, University of Minnesota, University of Nebraska, North Dakota State University, Oklahoma State University, South Dakota State University, and Texas A & M.

Ron Stammen, a faculty colleague in our TCU program, makes a fundamental point about distance education: "It's the pedagogy, not the technology!" This concept captures succinctly the dilemma and opportunity as educational administration departments plan for the delivery of programs to "distant students" using some form of technology. The primary concern always must be HOW teaching and learning can best be designed to accomplish desired outcomes. The deluge of more powerful and more complex instructional tools makes a proper balance difficult to find--a balance where professors don't "over-tech" or "under-tech" their courses. These conference Nygren Notes (in three categories: *defining*, *elements*, and *teaching tips*) reflect information and ideas about distance education synthesized and summarized from the Nebraska LearnShop.

Defining Distance Education from a College-Consortium Perspective

1. The current generation of distance education technologies is dominated by microcomputers, the Internet, and World Wide Web. (Virtual reality and artificial intelligence are on the horizon). The goal is to get people into the same electronic space so they can help one another learn. Reaching students not on campus is rooted historically in higher education's correspondence courses, but using posted mail has now given way to such electronic options as e-mail.

2. Higher education at end the of the century is "moving from a teaching to a learning paradigm". [Robert Anderson, Iowa State University].

3. Distance education technologies are becoming the norm for professional development and in-service training, particularly at the corporate level. It is estimated that 80% of Fortune 500 companies are using distance education techniques to increase access, decrease cost, and improve quality.

4. Adult learners need a convenient time and place (possible with distance education) and prefer learning from real-life situations. They believe the best "courses" often have no text but are the lessons of life. An effective technique for a professor is not to have the right answers, but the right questions.

5. A real issue in learning (distance education or not) is communications. The decision is to choose those tools of communication (high or low tech) which cause messages to flow efficiently and correctly. A primary societal shift in communications has been from readers to viewers. Some examples:

Print Medium THEN	Video Medium NOW
detailed (abstract)	brief (concrete)
issues (complex)	stories (simple)
thoughtful	emotional

Elements of Distance Education

1. Educational administration professors, if not already comfortable with such technological tools, need to quickly develop skills in the use of electronic mail (e.g., delivery of course materials, sending assignments, getting/giving feedback, listserv), bulletin boards/newsgroups (e.g., electronic discussion groups), downloading of course materials, interactive tutorials on the Web, and informatics--the use of on-line databases, library catalogs, and other websites to acquire information and pursue research. Handouts: "How to Cite Information from the World Wide Web" and "Copyright: A Sticky Issue on the Web."

2. An interesting and essential resource is A*DEC--a national distance learning consortium of state universities and land grant institutions providing high quality distance education programs and services via the latest and most appropriate information technologies. This organization, based at the University of Nebraska, should be the first listing on your "bookmarks" as you get seriously engaged with distance education. A*DEC has developed four Guiding Principles for Distance Learning:

a) Design active and effective learning.
b) Support the needs of learners.
c) Develop and maintain the technological and human infrastructure
d) Sustain administrative and organizational commitment

Check the web site for more detail on principles. [Web site: http://www.adec.edu]

3. All who use the Internet are aware that limited bandwidth, slow modems, and other system capabilities are hampering the efficient, timely use of this remarkable technology for instruction and research--particularly delivery of sound, video, and graphics. The system is dangerously overloaded and is getting worse every hour. The World Wide Web doubles every 45 days, a new home page appears every minute, and 73 commercial sites are added to the web each day on the average.

4. This problem has a solution. A group of about 100 universities is organizing Internet2 designed to develop the next generation of computer network applications to facilitate the research and education missions of higher education. Internet2 will enable member universities to develop and use certain bandwidth-intensive applications, such as desktop video and distance teaching. Internet2 will not take the place of the existing Internet. However, the university group had agreed to use only the new network for communication among themselves related to the goals of teaching and research.

Teaching Tips for Distance Education Professors

- A committee is searching for the ideal software vehicle for instructors and students to use in experiencing a positive and productive learning experience on the Internet. Eudora and Websites are email utilized, however the option being considered is having NDSU's Information Technology Service develop software rather than using FirstClass or Lotus Notes' LearningSpace.
- In deed, either system permitted a rather easy involvement of students in electronic group work (an approach superior to using listservs).
- When students are learning-on-line (often home-based) there is a need for each to practice and become comfortable with the hardware and software being used *well before* the actual course content begins.
- A great concern of many professors is that distance education (electronically offered academics) de-personalizes teaching and seldom should be entertained as a preferred instructional strategy.
- In many educational administration programs; students (and faculty members) now have weekly driving marathons.
- Clearly, distance education (with all of its potential) is not an option if students do not have access to the Internet.

Summary

The following information about the presenters provides a summary statement about this panel discussion held during the conference:

Ronald M. Stammen completed a two-year, USWEST Foundation funded project ($287,000) which focused on utilization of multimedia for K-12 teachers and professors. He is also involved as instructor-of-record and evaluator of USWEST/NDEA laptops initiative for 101 teachers across North Dakota, and Cass County School Improvement Coordinator for technology networking Professional Development and Curriculum Teams. He assisted Denny Van Berkum with basic team-based cooperative-online instruction for Ed Law.

Dennis W. Van Berkum is Chair of the Tri-College University Educational Leadership Program and Chair of the Moorhead State Universitys’ Faculty Technology Committee. He co-directed the collaborative with the University of North Dakota to jointly offer a new masters’ degree across North Dakota via the Interactive Video Network (IVN) and developed the Educational Leadership Program for Interactive Television Network in the central-northwestern region in Minnesota.

Burton N. Nygren is developing modules after attending a distance learning workshop involving eight universities hosted by the University of Nebraska-Lincoln. He has initiated research focusing on utilizing an electronic university for rural administrators. He teaches core program courses over the Interactive Video Network in North Dakota and over Moorhead State Universitys’ Interactive Television Network to NW Minnesota sites.

TECHNOLOGY, LEARNING, AND EQUITY ISSUES

Dennis W. Van Berkum
Associate Professor
Tri-College University/ Moorhead State University
vanberku@mhd1.moorhead.msus.edu

Burton Nygren
Assistant Professor
Tri-College University/North Dakota State University
nygren@plains.NoDak.edu

Sally I. Sologuk
North Dakota State University
sologuk@plains.nodak.edu

Abstract

Equity issues and the use of technology relates to much larger societal issues involving learning styles, gender, minority status, and socioeconomic factors that may be beyond the scope of most administrators and teachers duties to resolve. However, an ever increasing body of literature suggests that educational planners should consider these factors as they plan successful technology programs for all students. The issue of gender use is viewed from two perspectives, skills and attitudes. While women may have the same skills as men possess, they may be less apt to join in computer classes and computer generated discussion due to sociological expectations. Minority access and utilization has been directly associated to issues in school finance, policy, and school boundaries. Socioeconomic concerns are impacted by students ability to purchase computers outside the school and parental attitudes towards computer use. The purpose of this paper and discussion is to present some basic premises of learning and discuss the complexities of learning and instruction as it applies to equity issues when computers are used to deliver the curriculum.

TECHNOLOGY, LEARNING, AND EQUITY ISSUES

Dennis W. Van Berkum
Associate Professor
Tri-College University/ Moorhead State University
vanberku@mhd1.moorhead.msus.edu

Burton Nygren
Assistant Professor
Tri-College University/North Dakota State University
nygren@plains.NoDak.edu

Sally Sologuk
North Dakota State University
sologuk@plains.nodak.edu

Introduction

As computer usage in education continues to expand in schools, a variety of issues should be considered by teachers and administrators. These issues are derived from basic components of instruction and learning as they apply to student performance and the effectiveness of technology. Equity issues involving gender, race, or social-economic status may impact the ability for students to use virtual learning. Regardless of the issues teachers and administrators must become sensitive to theses issues as they develop schools, curricula, and classes which use computers for instruction. The following discussion offers some insight into the debate over the effectiveness of computer in the classroom and the computers as a method of delivery for learners.

Papert (1980) theorizes that the benefits of computers to aid children's learning and cognitive development using the Logo programming language. Further, he suggests that computers would become the vehicles for solving many of the problems in education. Through a technocentric approach [what may now be called the virtual classroom] to teaching and learning, teachers would be replaced by computers. This subject has come under discussion and debate by others (Becker, 1987; Davy, 1984; Maddux, 1989; Pea, 1983; Walker, 1987), resulting in the conclusion that more caution and deliberation should be used in the use of education.

About this same time period, Perelman (1990) called for vendors of computers to "seize the opportunity" to present computers as a means to replace teachers in the classroom. Package approaches such the Integrated learning systems (ILSs) were developed. Theses systems extensively use technology (particularly computers) in the teaching and the learning process. By integrating hardware, software, curriculum, and instruction into a package, students work independently to learn. The teacher's role is transformed from that of instructor to that of manager [facilitator] of instruction and creator of new curricular material. One only has to observe the number of independent study course that use the world wide web to instruct courses. The virtual classroom is one of the fastest growing phenomena in education across the Nation at all levels.

A national study, conducted by the Educational Products Information Exchange (EPIE) Institute of Brookvale (1990), suggests that this format to learning has met with favor by students, teacher and students. Teachers and administrator (96%) in the 24 schools who participated in the study supported the use of and recommended the implementation of similar systems. The major benefits to LISs most often cited by teachers and administrators were the individualization of instruction, the extensive reporting capabilities, and the completeness of content.

The results of the EPIE study (1990) offer enticements for all schools to enter the technological age and offer some computer instruction. However, this same study recommends that schools carefully plan for the use of these system and suggests that full implementation must proceed with caution. Teachers and administrator must consider the impact computer use has on all children. Administrators must ensure policies and procedures to protect the individual learning styles and economic capabilities of all students. Teachers must fully understand how these systems are integrated with other teaching activities, program of study, and the effects on individual students. Active Staff development programs should be developed to insure knowledge and implementation of appropriate practices.

The purpose of this panel discussion is to present some basic premises of learning and discuss the complexities of learning and instruction as it applies to equity issues when computers are used to deliver the curriculum.

Learning

Bigge (1982) defines learning as “an enduring change in a living individual that is not heralded by his genetic inheritance. Learning may be considered a change in insights, behavior, perception, or motivation or a combination of these; learning is always refers to some systematic change in behavior or behavior disposition that occurs as a consequence of experiences in some specific situation” (p. 1-2). How this concept is applied has resulted in many different interpretations of leaning. Most pretwentieth-century learning theories, that are taught and used today, are classified into three general categories namely: mental discipline; natural unfoldment or self-actualization, and apperception. Mental discipline means that learning consist of minds being disciplined or programs. The uses of drill and practice in common place. Students are expected to stay with the lesson. Rewards are based upon successful replication of the task. Punishment is administered with little hesitation using various physical and mental techniques. Natural unfoldment or self-actualization is the extreme opposite. In this theoretical concept, the teacher waits for the child to express a desire to learn. Apperception is a process of new ideas associating themselves with ones that have already constitute a mind. For example, reading is taught in a sequential manner where the teacher assigns students a set of rules to follow. Teachers are interest in making reading interesting and being sure that the right ideas are gleaned from the reading.

Twentieth century systematic learning theories are generally classified into two broad families. S-R (Stimulus-response) conditioning and cognitive theories. S-R conditioning is where learning is a change in observable behavior which is the result occurs through stimuli and responses becoming related to a set of mechanical principles. Cognitive theories are rooted in the Gestalt-field family. Learning is viewed as a process of gaining or changing insights, outlooks, or thought patterns. Gestalt teachers would help students get the ideas that surround the set of principles (Bigge, 1982).

Piaget (1971) epistemologies learning theory as a developmental sequence. Learning is viewed as the acquisition of knowledge through a set of developmental stages. For Piaget, the mental development of any child consists of a succession of three stages or periods, namely sensormotor, symbolic or pre-concrete-operational, and concrete operational. Each stage extends the preceding stage, reconstructs cognition on a new level, and comes to surpass the earlier stage. During preadolescence and adolescence the stage of formal operations emerges. Therefore, there is a need to consider learner readiness when using computer fir instructional purposes

The development of the theories of multiple intelligence, learning styles and brain-based learning has created another perspective that should be considered. Each of these explanations of learning styles brings an approach to teaching that focuses on how each student learns and describes the need to individualize instruction accordingly. Guild and Chock-Eng (1998) propose six common areas of overlap:

1. Each of the theories is learning and learning centered. Students are at the center of these schools and the energies of teachers is to help all student succeed. Curriculum is based upon student ability to comprehend and acquire knowledge.
2. The teacher is a reflective practitioner. Conversations about what works and what does not work are common. Adjustments are made to aid student individuality.
3. The student is a reflective practitioner. They are engaged in experimenting, creating, applying, and evaluating their ways of learning as well as interacting actively with the content and concepts they are studying.
4. The whole person is educated. Learning in the classroom is personalize by connecting the student's total life to the content of the curriculum.
5. The curriculum has substance, depth, and quality. Basic skills and other are treated seriously and within the context of the appropriate application. High expectation are the standard while standardization of curriculum and methodologies are avoided.
6. Diversity is promoted. Respect for similarities and differences is celebrated.

Technology learning

Tomei (1997) proposes that instructional technology offers a pedagogy for the futures. He suggests that instructional technology has " move to the forefront of education in a big hurry" (p. 56). In his article, Tomei offers the following as the most popular technologies are used to support classroom learning. With advancement in computer instruction, all approaches can be accomplished with computer assisted instruction. The approaches include:

1. Audio only
2. Audioconference
3. CBT (Computer-Based Teaching)
4. CDI (Computer Delivered Instruction)
5. CMI (Computer -Managed Instruction)
6. Computer Conference
7. Electronic Mail
8. Internet
9. Video Conference
10. Video Only

Further, Tomei (1997) matches these approaches to three paradigms: Enactive Classroom learning, Learning and Classroom Focus; and Presenting Student Learning. Enactive Classroom Learning is identified as the most efficient and effective form of interaction between teachers and students. This form of instruction attempts to match learning styles of the students with instruction from the sensormotor to the concrete operational to the abstract stages identifies by Piaget. The use of Enactive learning in the classroom is directly affected by the level of autonomy or readiness of the learner. Hands-on learning should match this progression where greater autonomy should result in greater levels of active learning. Audio only and video only provide for low levels of hands-on learning while rely on growth in learner autonomy. Audio and Video conferencing provide for more interaction and push the limits of learner autonomy. The Internet and Computer Conferencing heighten the use of both high interaction and high individual-learner independence. All methods need to be used to address all types of student learning styles.

Learning and Classroom Focus is a paradigm that consolidates three major schools of learning: behaviorism, constructivism, and humanism. Behaviorism offers that learning is evidenced by changes in behavior. The classroom is one that is teacher centered and the teacher presents facts and skills. Learning occurs via drill and practice. Computer assisted instruction has its roots in this principle and is enhanced with CMI and CBT.

The constructivist views learning as the continued development of schema where experiences are structure for individual knowledge. All form of technology are effect in aiding student to develop their metacognitive skills with the exception of CDI. Humanism is identified as a two step process where the acquisition of knowledge is followed by individual personalization. The class is free from threats and an abundant amount of resources are available for students to choose. Interpersonal skills are encourage by audio and video conferencing, with computer-aid conferencing and the Internet being the mots effective. Presenting Student Learning is based upon notion that students in earlier grades are best taught by concrete examples. All of the instructional technologies have application to some level of development of students. As a student develops, they can move to more sophisticate means of instruction.

Equity issues

With the on-going development of courses for computers, virtual classroom developers and delivers should consider how a number of critical issues related to equity are emerging.   For the purposes of this discussion, equity refers to the qualitative properties (issues of content or curriculum and will be interchanged with equality (issues measured by ratios).

Much of the literature concerning the issue of gender differences in computer instruction is inconclusive and complicated. Both men and women demonstrate the ability to use computer equipment every day. But, research indicates that a difference exist in the approach to the use of computer during particularly the formative years. Computer acceptance by females and males is related to larger societal issues and sex stereotyping. Generally, the literature in this area is comprised of two areas: one dealing with the performance in computer courses, and the other deal with attitudes to computing (Picciano, 1998).

The results of studies to determine gender differences with student performance in inconclusive. Picciano (1998) suggests that a possible reason for the differences in performances can be found in the use of computers. Computer skill tends to increase as one use the computer more. If boys use the computer more than girls, than they would show greater higher levels of performance and greater levels of competence, particularly in upper grades. While this may be true, studies conduct using Logo skill was inconclusive at the earlier years. Flock, Simpson, and Reid (1987) reported differences that favored boys in kindergarten, first grade, and second grade. Schaefer and Sprigle (1989) found no differences. Campbell, Fein, Scholnik, Schwartz, and Frank (1986) may be closest to the reason. They found differences in programming style but not in programming mastery.

Although research comparing gender differences has been inconsistent, research comparing attitudes has been more consistent. Studies conducted where computer usage is an option is more favorably viewed by males than females. Males tend to choose electives, join computer clubs, and major in computer sciences. Current thoughts about this difference tend to focus on larger societal factors such as: parental influence, subject stereotyping, peer influence, and access to computers (Picciano, 1998). For example, a study conducted by Shashaani (1994) of 1730 high school students conclude that significant gender differences in computer interest, computer confidence, and gender-stereotyped view exist. Further, parental attitudes and influences are directly associated with student attitudes about computing.

Bonnell (as cited in Blumenstyk, 1997) offers another way to look at this dilemma. She offers that "the goal of many courses is to merely let students master material. She suggests that women do have different learning styles than men [Feminist pedagogy]. Women react better than men to certain teaching approaches. For example, women have been socialized to speak less often than men in class. Therefore, they may be more reluctant to participated in computer classes. The way questions are constructed to invite a collegial discussion become very important. If the class become a debate women may not contribute at all.

Sanders and Stone (1986) suggest the following strategies to consider for gender issues:

1. Require students to take certain computer courses because girls tend to enroll in much smaller numbers than boys if courses are optional.
2. Expand computer curricula beyond mathematics and science courses.
3. Integrate computing into the regular academic program.
4. Educate parents to ensure that the home environment does not contribute to stereotyping.
5. Educate staff so that teacher are aware of the issue.
6. Establish positive role models in the schools.
7. Review and eliminate software and computer literature that might contain stereotypical characterizations of depiction.

For more information on sex equity and computing : Women's Action Alliance (Computer Equity Program, 370 Lexington Ave. Suite 603, New York, NY 10017) and the Computer Equity Training Project of the U.S. Department of Education (555 New Jersey Ave. NW, Washington, DC. 20208-5646.

Socio-economic and minority issues

Computer usage requires financial resources. There appears to be a direct relationship between the amount of money a school has and the school's ability to purchase and participate in computer utilization. Thus, schools with fewer financial resources have less money offer less opportunities for students to develop skills and positive attitudes toward computer literacy. The school with the least amount of money tend to be those schools that have greater minority populations and greater populations of low income families. In addition, urban and larger schools offer less access to computer than smaller schools. Quality Education Data (1991) conducted three national studies concerning computer usage. The study stated in the conclusions that the acceleration of computers did take place in schools with the greatest percentage of increase in suburban area where most white population are located and where leadership is most prevalent and progressive. If this finding is true, and given that computer attitude, computer knowledge, and computer skill are related to computer access, then schools with the greatest minority populations have the least number of computers. Therefore, attitude, knowledge, and skill will be less in minority populations. The same tendency exist in larger school. School size and location are likely to impact students ability to gain positive attitudes, increased knowledge, and adequate skill in computers (Picciano, 1998).

The following strategies may be considered with issues race and socioeconomic:

1. Leaders need to focus attention on groups who do not have access to computers due to financial restrains in the school district.
2. Parents are integral in developing attitudes towards student attitudes. Educate parents in the use of computer instruction in your school for school work so the computer enriches the learning experience.
3. Carefully select software that in not racially, culturally, or economically bias.
4. Consider strategies that do not isolate groups from one another instead of bringing them together.
5. The more computer based instruction is implemented, consideration should be made to providing more access to equipment beyond the normally scheduled classroom. Ones needs to become aware of the number of students who have access to computers at home and consider doing something for those who do not have access.
6. Leadership appears to be instrumental to computer implementation in the classroom (Picciano, 1998).

Conclusions

The use of computers in education is here to stay and offers a wide array of options to improve what we teach, when we teach, why we teach, where we teach, and how we teach. Computerized instruction offers a variety of methods to address equity issues in education. However in our haste to reach for greater opportunities, we must remain conscious of the effect this technology has on learning for all student. We need to understand how technology effects leaning and the learning process so we do not eliminate certain student populations. Discriminatory practices must be considered when developing classes, courses, and programs. Access to computer is essential for all students. Carefully constructed lessons and programs need to address the individual learning styles so all students can gain from technologies contributions to education.

References

Becker, H. J. (1994). Analysis and trends of school use of new information technologies. Irvine: Department of Education, University of California, Irvine.

Bigge, M.L. (1982). Learning theories for teachers. 4th ed. New York: Harper and Row.

Blumenstyk, G. (1997, October 31). A feminist scholar questions how women fare in distance education. The Chronicle of Higher Education.

Campbell, P.F., Rein, G.G., Scholnidk, E.K., Schwartz, S.S., & Frank, R.E. (1986). Initial master of the syntax and semantics of Logo positioning commands. Journal of Educational Computing Research. 3, 435-442.

Davy, J. (1984). Mindstorms in the lamplight. Teachers College Record. 85(4), 549-558.

Guild, P.B. & Chck-Eng, S. (1998). Multiple intelligence, learning style, brain based education: Where do the messages overlap? School in the Middle Theory to Practice . 7(4), 38-40.

Maddux, C. (1989). Logo: Scientific dedication or religious fanaticism in the 1990's. Educational Technology. 29(2), 18-23.

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. New York: Basic Books.

Pea, R.D. (1983). Logo programming and problem solving (Technical Report No. 12). New York: Bank of Street College of Education, Center for Children and Technology.

Perelman, L. (1990, October 8). Can technology effectively replace human teachers? Computerworld. p. 25.

Piaget, J. (1971). Psychology and epistemology. New York: Grossman.

Picciano, A.G. (1998). Technology, learning, and equity issues. Educational Leadership and Planning for Technology 2nd ed. Upper Saddle River, NJ: Prentice-Hall Inc. p. 37-55.

Picciano, A.G. (1991). Computers, city, and suburb: A study of New York City and Westchester County public schools. The Urban Review 23(3), 93-109.

Quality Education Data. (1995). Education market guide and mailing list catalog. 1995-1996. Denver: Author.

Sanders, J.S., & Stone, A. (1986). The neuter computer. New York: Schuman.

Shashaani, L. (1994). Socioeconomic status, parent's sex-role stereotypes, and the gender gap in computing. Journal of Research on Computing in Education 26(4), 433-351.

Tomei, L.A. (1997). Instructional technology: Pedagogy for the Future. The Journal. 29(5), 56-59.

Walker, D.F. (1985). The role of gender in computer programming learning processes. Journal of Educational Computing Research. 1, 441-458.

Using Hash Tables To Investigate Zipf's Law

Dr. John M. Weiss and Dr. Roger W. Johnson
Department of Mathematics and Computer Science
South Dakota School of Mines and Technology
501 East St. Joseph Street
Rapid City, SD 57701-3995
605-394-6145   jweiss@silver.sdsmt.edu

Abstract

Zipf's Law is a statistical curiosity describing the relationship between word rank and word frequency in text. Zipf's Law states that word rank multiplied by word frequency is roughly a constant. For most texts, regardless of author or language, this turns out to be true. In fact, in a slightly modified form, Zipf's Law even holds for random text.

We have developed an efficient algorithm for empirical studies of Zipf' s Law. With an increasing number of texts available in electronic form on the Internet (e.g., Project Gutenberg), it is now practical to verify Zipf's Law for a large number of text files. It is also interesting to test Zipf's Law on random text.

Our approach is based on the use of hash tables, and makes an excellent case study in a Data Structures class. A sample programming assignment is presented.

Using Hash Tables To Investigate Zipf's Law

Dr. John M. Weiss and Dr. Roger W. Johnson
Department of Mathematics and Computer Science
South Dakota School of Mines and Technology
501 East St. Joseph Street
Rapid City, SD 57701-3995
605-394-6145   jweiss@silver.sdsmt.edu

Introduction to Zipf's Law

Charles Dicken' s *A Christmas Carol* contains 4,296 distinct words with a total word count of 28,479 words. The word ' the' occurs 1,552 times and is the most frequently occurring word. Assign it a rank of 1. The word ' and' occurs 1,047 times and is the next most common word. Assign it a rank of 2. Continue in this manner to assign ranks to words based upon their frequency. The tenth most common word, for instance, is ' his' which occurs 416 times. Assign this word a rank of 10.

Zipf' s Law [Zipf, 1935; Zipf, 1949; apparently first noted by Estoup, 1916] is an observation that for most texts, regardless of author or language, rank multiplied by frequency is roughly a constant. Table 1 shows selected values of rank (in multiples of roughly 50 to keep the Table from getting too long, along with the ten smallest and ten largest ranks), frequency, and their product for *A Christmas Carol*. Zipf' s Law typically holds best when the rank values are not too small or too large.

To completely explain the entries which appear in Table 1, it should be noted that the listed ranks are, in fact, average ranks. This is a consequence of some words having the same frequency. To illustrate, the words ' mind' , ' towards' , and ' yes' each occur 19 times, and 195 words in *A Christmas Carol* occur more frequently than 19 times. To assign a rank to the three words ' mind' , ' towards' , and ' yes' , take the average (197) of the next three available ranks (196, 197, and 198).

Rank	**Frequency**	**Rank*Frequency**	**Number of Words with this Frequency**
1.0	1552	1552.0	1 (the)
2.0	1047	2094.0	1 (and)
3.0	690	2070.0	1 (a)
4.0	667	2668.0	1 (to)
5.0	648	3240.0	1 (of)
6.0	516	3096.0	1 (in)
7.0	508	3556.0	1 (it)
8.0	474	3792.0	1 (he)
9.0	419	3771.0	1 (was)
10.0	416	4160.0	1 (his)
.	.	.	.
.	.	.	.
50.0	86	4300.0	1 (my)
101.5	39	3958.5	4 (every, never, some, such)
148.5	27	4009.5	4 (back, can, quite, young)
197.0	19	3743.0	3 (mind, towards, yes)
250.0	15	3750.0	15 (better, bless, . . ., world, years)
301.5	12	3618.0	22 (called, city, . . ., tried, wish)
.	.	.	.
.	.	.	.
351.5	10	3515.0	38 (afternoon, . . ., wonderful)
384.5	9	3460.5	28 (along, . . ., wife)
421.0	8	3368.0	45 (bear, . . ., within)
474.0	7	3318.0	61 (above, . . ., you’ re)
544.5	6	3267.0	80 (ah, . . ., wonder)
639.5	5	3197.5	110 (ago, . . ., wretched)
776.0	4	3104.0	163 (account, . . ., yourself)
1023.5	3	3070.5	332 (accompanied, . . ., yours)
1549.0	2	3098.0	719 (able, . . ., yonder)
3102.5	1	3102.5	2388 (abed, . . ., zeal)

Table 1: Illustration of Zipf’s Law for *A Christmas Carol*

Note that

$$rank \cdot \ frequency = C$$

is equivalent to

$$ln(rank) + ln(frequency) = K$$

so that if Zipf’ s Law holds, then a plot of*ln(rank)* vs. *ln(frequency)* should yield a linear pattern with slope -1. Here is such a plot for *A Christmas Carol*:

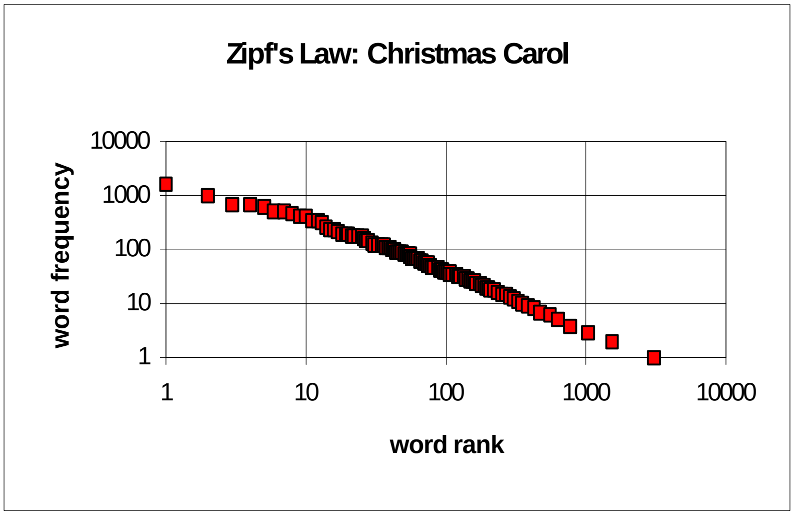


Figure 1: Zipf' s Law for *A Christmas Carol* (Charles Dickens).

The rank-frequency version of Zipf' s Law given above has a rough equivalent. Letting *D* be the number of distinct words in a text, the number-frequency version of Zipf' s Law may be stated as follows [Tuchinsky, 1980]:

The number of words which appear exactly once is $\frac{D}{(1)(2)}$;

The number of words which appear exactly twice is $\frac{D}{(2)(3)}$;

.
.
.

The number of words which appear exactly n times is $\frac{D}{(n)(n+1)}$.

For *A Christmas Carol*, the number of distinct words is *D = 4,296*. The number-frequency version of Zipf' s Law predicts 2,148 words will appear once, 716 words will appear twice, 358 words will appear three times, etc. compared to the actual values of 2,388, 719, 332, etc., respectively.

Zipf's Law for random text and other things

Various stochastic models have been presented which have Zipf' s Law and generalizations such as $frequency \cdot rank^q = constant$ as a consequence (see, for instance, [Read, 1981] and the references in [Zörnig and Altmann, 1995]), but the derivations tend to be rather involved. One rather simple model, which has as its consequence a Zipf-type Law, was considered by [Miller, 1957]. He imagines a monkey hitting keys of a typewriter subject to the following conditions:

1. She hits the space bar with chance *p*, and the other *A* keys at random with collective chance *1- p*. (For English text we may take *A* to be 26 and *p* to be about 0.18.)

2. She never hits the space bar twice in a row.

Miller shows, using an elementary probability argument, that

$$P(w) \cdot \ (rank(w) + c)^{(1 + d)} = constant$$

where *P(w)* is the probability of a particular word *w*, *rank(w)* is its rank,

$$\mathrm{c} = \frac{\mathrm{A}+1}{2(\mathrm{A}-1)} \quad \text{and} \quad \mathrm{d} = -\frac{\ln(1-\mathrm{p})}{\ln(\mathrm{A})} .$$

Consequently, estimating *P(w)* by its frequency divided by the total number of words, it should be approximately true that

$$frequency(w) \cdot \ (rank(w) + c)^{(1 + d)} = constant$$

For *A = 26* and *p = 0.18*, we find that *c = 0.54* and *d @0.0609*. After rounding things a bit, we end up with the frequency-rank version of Zipf’ s Law; namely, that frequency multiplied by rank is roughly a constant.

Zipf-type Laws hold in a variety of settings, not just for word frequencies in texts. A plot of *ln(population size)* versus *ln(rank)* of U.S. cities, for example, reveals a linear plot. (New York City has rank 1, Los Angeles has rank 2, etc.; population data is available from [U.S. Census Bureau, 1995]). For a description of some of these other areas where Zipf-type Laws hold, see [Simon, 1989].

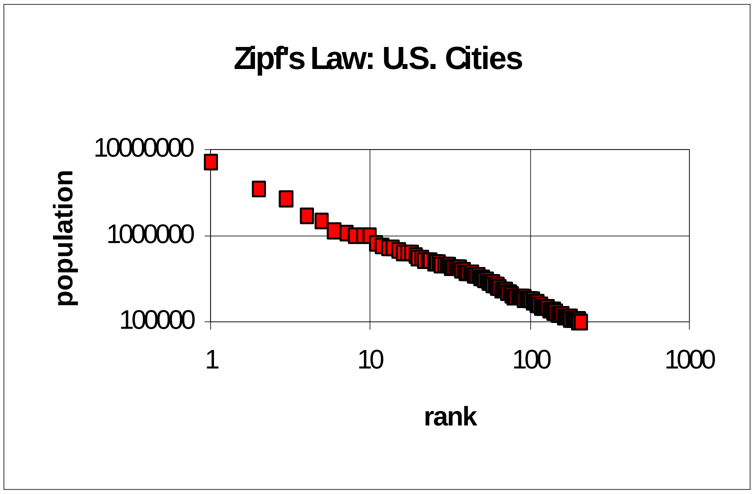


Figure 2: Zipf’ s Law for U.S. cities.

[Simon, 1989] and [Tuchinsky, 1980] contain very readable accounts of Zipf’ s Law for those who wish to investigate further. See [Chen, 1992] for background on Zipf’ s Law which has more of a computer science flavor.

Methodology

To test Zipf's Law, a word concordance is constructed from a text file. A *word concordance* is a listing of the words and their frequencies of occurrence in the text. The word concordance is then sorted in descending order of frequency. This allows words to be ranked, as described above. Finally, the rank-frequency relationship can be examined.

Constructing the word concordance efficiently takes a bit of effort. Characters must be read from a text file, and parsed into words. Each word must be stored, together with its frequency of occurrence, in a data structure. If the word has already been encountered, its frequency is incremented. Otherwise it is a new word, and must be inserted with a frequency of 1.

There are several possible choices for the data structure in which to store words and frequencies, some better than others. The data structure must support rapid searching, otherwise word lookup takes too long. The data structure must also support rapid insertion of new words. This eliminates, for example, maintaining a sorted table of words and frequencies. Although an alphabetically-sorted word array allows for rapid binary search [Weiss, 1996], word insertion takes far too long.

A better choice is a height-balanced search tree [Weiss, 1994], such as an AVL-tree or a B-tree. These structures allow finding a word in *O(log N)* time (where *N* is the number of words stored in the tree), and inserting a new word in *O(log N)* time. However, height-balanced trees are complicated to program, and require significant space overhead for pointers. Since the search tree is ordered alphabetically (to quickly find words), it will require more time and space overhead to sort the words by frequency. Furthermore, we can do better than *O(log N)*.

Hashing

A *hash table* is the data structure of choice for this problem. Hash tables are data structures that allow finding, inserting, and deleting items in *O(1)* average time [Knuth, 1975; Tenenbaum et al., 1990; Gonnet and Baeza-Yates, 1991; Weiss, 1994]. Although hash tables do not support all the operations of sorted lists or search trees (for example, finding the *next* item in a list is not supported), there is no data structure that is faster at the find, insert, and delete operations. For this particular application, only efficient find and insert operations are initially required. Once all the words and frequencies are inserted in the hash table, it can be sorted as a post-processing step in *O(N log N)* time.

It is straightforward to prove that *O(log N)* is a lower bound on finding an item in a list, when the basic operation is comparing two items [Weiss, 1994]. Hashing, however, does not work by comparing two items. Instead, a *hash function* is used to map a search key directly into an address in a hash table [Knott, 1975; McKenzie et al., 1990]. The same hash function is used to find items, insert items, and delete items in the hash table.

For example, consider a hash table of size *M*. Let us assume that there are *N* items (*N* < *M*) to be inserted into this table. If the search keys are integers, such as Social Security numbers, they can be converted into hash table addresses by taking the number modulus the table size. This would correspond to the following hash function:

```
        hash(key) = key % M;
```

If the search keys are strings, such as names, the ASCII values of the characters can be summed, and the hash function could return this sum modulus the table size:

```
        int sum = 0;
        for (int i = 0; i < strlen(key); i++)
                sum += key[i];
        hash(key) = sum % M;
```

Neither of these represent particularly good hash functions, but they suffice to give the basic idea: convert a search key into an integer in the range *[0, M-1]*. A good hash function has two properties: it is efficient *O(1)* to compute, and it distributes keys uniformly throughout the hash table, in order to minimize collisions. A better hash function for strings is the following, which converts the string to a base-26 number modulus the table size [Weiss, 1994]:

```
        int sum = 0;
```

```
for (int i = 0; i < strlen(key); i++)
      sum = (sum * 26 + key[i]) % M;
hash(key) = sum;
```

It is usually best to make the table size *M* a prime number.

Collisions are the central problem in hashing [Knuth, 1975; Tenenbaum et al., 1990; Weiss, 1994]. A collision occurs when two different keys map to the same table address:

```
      hash(key1) == hash(key2)
```

Collisions are virtually certain to occur. First, the number of possible keys is typically very large. To ensure that no collisions will occur, a hash table with at least that many entries is required. For example, consider using Social Security numbers as search keys. The total number of Social Security numbers is 999,999,999. It is generally impractical to allocate a hash table capable of storing 1 billion items! The situation is even worse if last names are used as search keys. Even if names are restricted to a maximum of ten letters, there are still 26^{10} possible last names, requiring a hash table of approximately 141 trillion slots.

Even when the number of possible keys is small, collisions are far more likely to occur than one might expect. The so-called *Birthday Paradox* [Feller, 1968; Weiss, 1994] illustrates this phenomenon. In a room with 23 people, the likelihood is greater than 50% that two people have the same birthday. (This is counter-intuitive to most people. After all, there are 365 possible birthdays.) This situation is analogous to a hash table that is only 6% full (23 items in 365 slots). Even when a hash table is quite empty, the likelihood of collisions is high. Allocating a very large hash table does not solve the problem of collisions.

There are two basic collision resolution schemes in hashing, known as open addressing (or closed hashing) and separate chaining (or open hashing) [Knuth, 1975; Maurer and Lewis, 1975]. In*chaining*, each slot in the hash table is a pointer to a linked list of all the items that hash to that particular address. Chaining is reasonably efficient (there is some space overhead for the linked list pointers), performs well in practice, and allows for storing more items in the hash table than the number of slots (N ‡ M).

Unfortunately, chaining poses some problems for studying Zipf's Law. After inserting all the words and counting their frequencies, the hash table must be sorted in descending order of frequency. Having to deal with the many small linked lists of chaining is impractical. Instead, we consider open addressing.

In *open addressing*, a *probe sequence* [Knuth, 1975; Maurer and Lewis, 1975] of table addresses is specified. During insertions, try the first address on this probe sequence, then the second, then the third, etc., until an empty slot is located, and the new item can be inserted. During find operations, try the first address, then the second, then the third, etc., until the item is found (success), or an empty slot is hit in the hash table (failure).

There are several popular probe sequences. The simplest is *linear probing* [Knuth, 1975; Maurer and Lewis, 1975], which uses the probe sequence

```
(h(k) + i) % M                               (i = 0,1,2,3,...)
```

In other words, try the next slot in the table, then the next, etc., being sure to wrap around to the start if the end of the table is reached.

The problem with linear probing is *primary clustering* [Knuth, 1975; Maurer and Lewis, 1975]. The *load factor*, defined as N / M, gives a measure of how full the hash table has become. As the load factor gets large, the average number of probes increases, reducing performance. Primary clustering describes the tendency to build up large clusters of items in the table, leading to very long probe sequences. This can significantly reduce performance.

A superior technique is *quadratic probing* [Maurer and Lewis, 1975; Weiss, 1974]. In quadratic probing, the probe sequence is

$$(h(k) + i^2) \ \% \ M \qquad\qquad (i = 0,1,2,3,...)$$

After trying the first address h(k), probe h(k)+1, h(k)+4, h(k)+9, etc. This is almost as efficient to implement as linear probing, and breaks up primary clustering quite well. However, to guarantee that every slot in the table will be probed, the table size should be prime, and the load factor kept to under 0.5.

Double hashing [Weiss, 1974] defines yet another probe sequence:

$$(h_1(k) + i * h_2(k)) \ \% \ M \qquad\qquad (i = 0,1,2,3,...)$$

where $h_1(k)$ is a primary hash function, and $h_2(k)$ is a secondary hash function. Double hashing breaks up clustering even more effectively than quadratic probing. In practice, however, the overhead incurred by the second hash function is not always worthwhile.

Implementation

To investigate Zipf’ s Law, we wrote a program in C++ that implements the following algorithm:

1. Read the text file line-by-line, parsing each line into words.

   - OR -

   Generate the desired number of random words.
2. Hash each word into the hash table. If the word is already in the table, increment its frequency. If it is a new word, insert it in the table[*] with a frequency of 1. This takes *O(N)* time for *N* input words.
3. Compact the hash table, moving NULL entries to the upper end of the array. The quicksort partitioning scheme will do this in *O(M)* time for table size *M*.
4. Sort the first *N* entries of the hash table in descending order of frequency, and alphabetically within frequency groups. Any good sort will do this in $O(N \times logN)$ time.
5. Output rank-frequency data to a disk file for plotting.

Words in the text file words are parsed using the *strtok()* function in the standard string library of C. A word is defined as a string of letters, possibly containing digits and single quotes (to allow for contractions). White space and punctuation marks other than the single quote are considered word separators. Words are converted to lower case prior to insertion in the hash table.

The program implements a hash table of words and frequencies. The hash function is similar to the one listed above for strings, using quadratic probing to resolve collisions.

Determining the hash table size is an interesting problem. For quadratic probing, the hash table size *M* should be a prime number at least twice as large as the number of distinct words *N* in the text file. Choosing an arbitrary large prime *M* is wasteful of space; ideally, *M* will be the smallest prime number such that $M > 2N$. Unfortunately, *N* is not known until the text is processed, so some guesswork is involved.

[*] If initial hash table size proves to be too small, rehash the words into a larger table.

For input text files, the hash table size *M* is initially guessed to be roughly 5% of the input file size in bytes, with a minimum of 16001 slots and a maximum of 64001 slots. For random text, the hash table size is initially guessed to be approximately half as large as the number of random words to be generated. These values were determined empirically to perform well in practice on a wide variety of texts. However, during text processing, we may find that the initial hash table size is too small. This occurs when a new word is encountered in the text, and the number of distinct words *N* becomes larger than *0.5M*. When this situation occurs, rehashing is required.

In *rehashing*, a new hash table of roughly twice the size of the old hash table is allocated. Then the words and frequencies in the old hash table are rehashed into the new hash table. Finally, the old hash table is deallocated. This process takes *O(M)* time and *O(M)* extra space.

Note that we cannot simply copy entries from the old hash table into the same slots in the new hash table. Since the new hash table will have a new hash function (which computes a value modulus the new table size), each entry must be rehashed into the new hash table.

Processing of the input text can be now continue as before. Rehashing may take place as many times as needed.

After hashing all the words into the hash table, the table is compacted by moving all filled slots to one end of the table, and all empty slots to the other. The quicksort partitioning scheme [Hoare, 1962] is an efficient *O(M)* way to handle this:

```
int i = 0, j = M - 1;
while (i < j)
{
      // find next empty slot on left
      while (Table[i] is not empty) i++;
      // find next nonempty slot on right
      while (Table[j] is empty)     j--;
      // swap entries
      Swap(Table[i], Table[j]);
}
```

Now the *N* distinct words and their associated frequencies are stored at the lower end of the table, suitable for sorting. The built-in *qsort()* function in the C standard library is used for sorting the table in descending order of frequency, and alphabetically within frequency groups. The *qsort()* routine is a generic implementation of quicksort, and hence runs in *O(N log N)* time [Hoare, 1962; Knuth, 1975; Weiss, 1994]. Any reasonably efficient sorting routine may be used instead.

After sorting the first *N* items in the hash table, two files are output. The first output file is a table of *rank* vs. *frequency*, suitable for plotting. The second output file contains the word concordance itself, which is not strictly necessary for verifying Zipf's Law, but is convenient for error-checking and other applications.

Results

The program to investigate Zipf' s Law was written in C++ and compiled using DJGPP, a DOS port of the GNU C/C++ compiler [Delorie, 1997]. DJGPP produces efficient 32-bit code via a DOS extender, allowing large arrays (up to available RAM) to be allocated. Tests were run on a Pentium P133 computer with 32M RAM and a reasonably fast (10 msec access) hard disk.

Zipf' s Law appears to hold for a wide variety of texts. Rank-frequency plots for selected texts follow. These texts and many others may be obtained electronically through the Gutenberg Project [Gutenberg Project, 1997]. Such plots are much easier to look at than numerical summaries such as those given in Table 1.

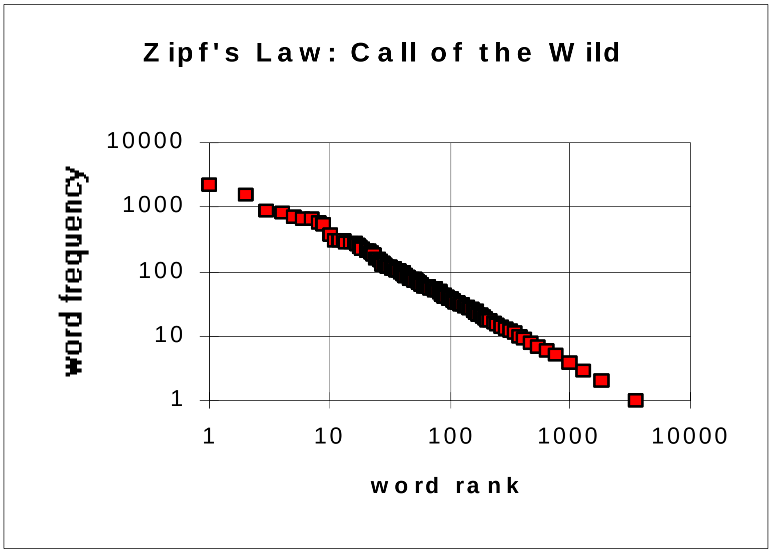


Figure 3: Zipf’ s Law for *Call of the Wild* (Jack London).

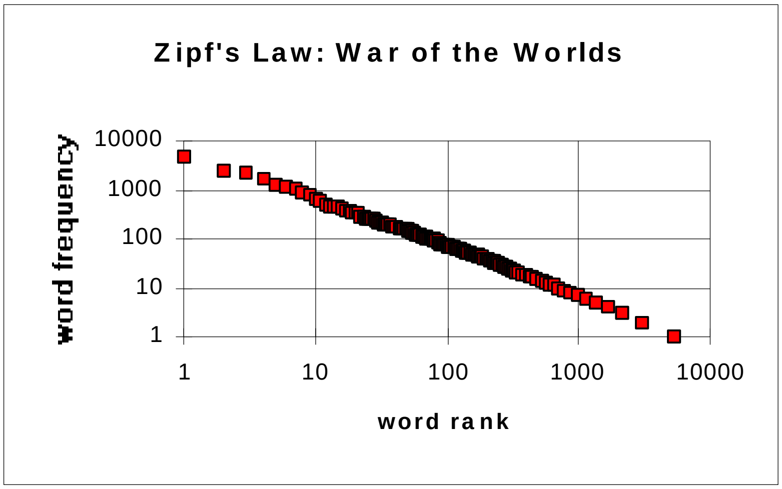


Figure 4: Zipf’ s Law for *War of the Worlds* (H.G. Wells).

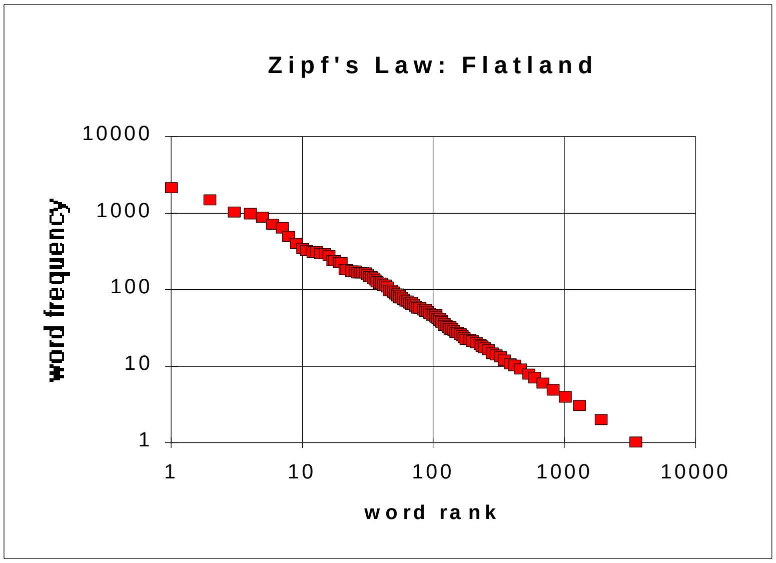


Figure 5: Zipf' s Law for *Flatland* (Edwin Abbot, 1884).

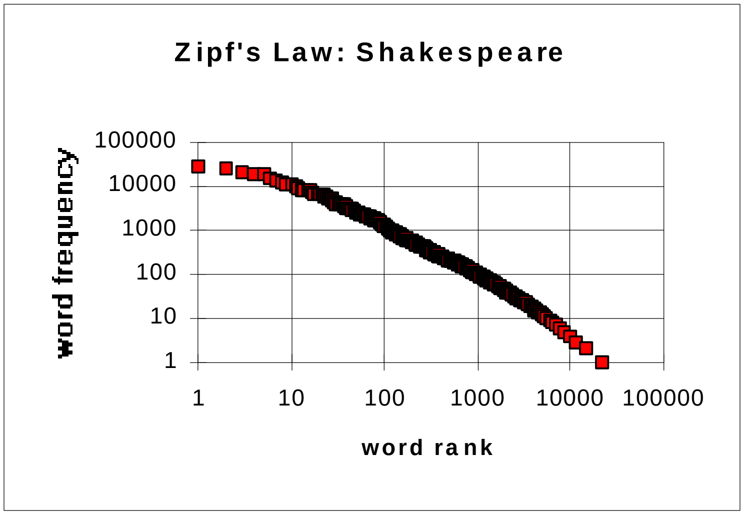


Figure 6: Zipf' s Law for *The Complete Works of Shakespeare*.

The validity of the Zipf-type Law for random text is less obvious, however. Plots of $frequency(w) \cdot (rank(w) + c)^{(1 + d)}$ show a "sawtooth" rather than a horizontal line, as predicted by the model. This phenomenon is illustrated in Figure 7.

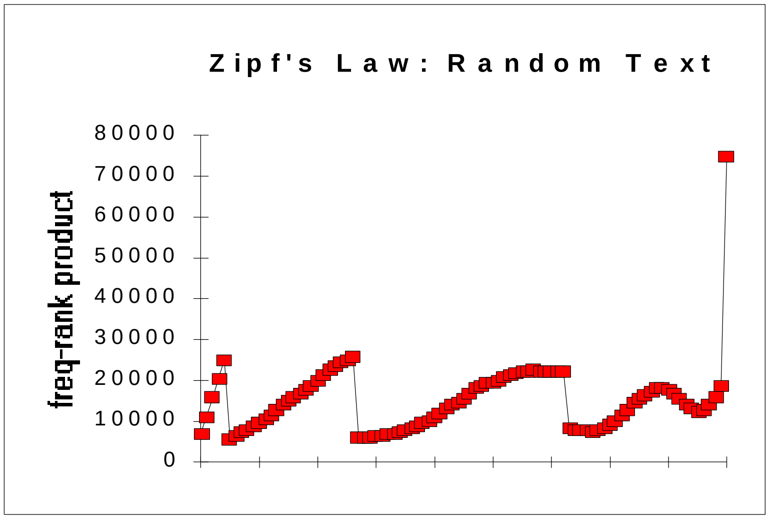


Figure 7: Zipf' s Law for random text (*A = 5* and *p = 0.18*).

What causes this odd behavior? For random words generated from an alphabet of *A* = *5* letters and space probability *p* = *0.18*, the most common words are the five one-letter words *a*, *b*, *c*, *d*, and *e*. Following these are the twenty-five two-letter words *aa*, *ab*, etc. Ideally, all one-letter words would have exactly the same frequency of occurrence in random text, and each would be assigned the same rank. In practice, however, the frequencies of the one-letter words are roughly (but not exactly) the same, and they are ranked 1, 2, 3, 4, and 5. This causes the *frequency-rank* products to differ significantly, and accounts for the sawtooth appearance of the plot.

Performance

Hashing allows rapid processing of very large text files. For example, the complete works of Shakespeare (5,582,655 bytes, 904,574 words, 26,162 distinct words) are processed in under 10 seconds on a Pentium P133 computer with 32M RAM and a reasonably fast hard disk. Most reasonably-sized text files are processed in under 1 second. The average number of probes is typically 1.0 to 1.1, regardless of whether linear or quadratic probing is used.

Random text requires large hash tables, since relatively few random words occur many times. Once again, little difference was noted between linear probing and quadratic probing for a good hash function and light load factors (less than 0.5). For example, with an alphabet of 26 letters and space probability of 0.18, the average number of probes for linear probing was 1.3, versus 1.2 for quadratic probing. Either approach took took approximately 2.7 seconds to process 100,000 random words.

However, with a hash function that is ill-suited for processing random text, the differences between linear and quadratic probing were dramatic. Using the built-in hashpjw() routine of the GNU C library [Delorie, 1977], the average number of probes for linear probing was 317.8, which took 28.4 seconds. The average number of probes for quadratic probing was 6.1, which took only 3.0 seconds. It is clear that a combination of a poorly-chosen hash function together with linear probing may lead to significant performance degradation in hashing.

Conclusions

This paper describes an empirical investigation of Zipf' s Law. For a wide variety of electronic texts (available on the Internet via Project Gutenberg), the rank-frequency relationship appears to hold true. The Zipf-type law for random text also appears to be valid in practice, although the results show an interesting sawtooth effect that is not predicted by theory.

In order to test Zipf' s Law, a word concordance must be constructed. Our implementation is based upon the use of a hash table. This makes an excellent case study for a Data Structures class in an undergraduate Computer Science program. A sample programming assignment is presented in the Appendix. Most students successfully completed this assignment in two weeks time, and learned many important aspects of hashing from it.

The programming assignment and our implementation of the program are available electronically on the Internet, at the URL *ftp://ftp.sdsmt.edu/pub/sdsmt/jmw.*

References

Chen, Ye-Sho (1992) "Mathematical modeling of empirical laws in computer applications: A case study", *Computers & Mathematics with Applications*, vol. 24, no. 7, pp. 77-87.

Delorie, D.J. (1997) DJGPP documentation, available at the URL http://www.delorie.com.

Estoup, J.B. (1916) *Gammes Sténographiques*, 4th edition, Paris.

Feller, William (1968) *An Introduction to Probability Theory and Its Applications, Vol. 1*, Wiley, New York, p. 33.

Gonnet, G. and Baeza-Yates, R. (1991) *Handbook of Algorithms and Data Structures*, second edition, Addison-Wesley, Reading, MA.

Gutenberg Project (1997) Electronic texts of a variety of classic works may be found at the URL *ftp://uiarchive.uiuc.edu/pub/etext/gutenberg*, University of Illinois at Urbana-Champaign.

Hoare, C.A.R. (1962) "Quicksort", *Computer Journal 5*, pp. 10-15.

Knott, G. (1975) "Hashing Functions", *Computer Journal 18*.

Knuth, Donald E. (1975) *The Art of Computer Programming, Vol.3: Sorting and Searching*, second printing, Addison-Wesley, Reading, MA.

Maurer, W. and Lewis, T. (1975) "Hash Table Methods", *Computing Surveys 7*, pp. 5-20.

McKenzie, B., Harries, R. and Bell, T. (1990) "Selecting a Hashing Algorithm", *Software - Practice and Experience 20*, pp. 209-224.

Miller, George A. (1957) "Some effects of intermittent silence", *The American Journal of Psychology*, vol. 70, pp. 311-314.

Read, Campbell B. (1981) "Zipf' s Law", *Encyclopedia of Statistical Sciences*, Wiley, New York, pp. 674-676.

Simon, Herbert A. (1989) "The sizes of things", appears in Tanur, Judith M. et.al., editors (1989), *Statistics: A Guide to the Unknown*, third edition, Wadsworth & Brooks/Cole, Pacific Grove, CA, pp. 142-150.

Tenenbaum, A., Langsam, Y. and Augenstein, M. (1990)*Data Structures Using C*, Prentice-Hall, Englewood Cliffs, NJ, pp. 454-500.

Tuchinsky, Philip (1980) “Zipf’ s Law and his efforts to use infinite series in linguistics”, Modules in Undergraduate Mathematics and Its Applications, UMAP Module 215, COMAP, Arlington, MA (617) 641-2600.

U.S. Census Bureau (1995) Data available at the URL http://www.census.gov/population/ estimates/metro-city/sc100k94.txt, and also listed in *Statistical Abstract of the United States 1996*, 116th edition, U.S. Government Printing Office, Washington, DC, pp. 44-46.

Weiss, Mark A. (1994) *Data Structures and Algorithm Analysis in C++*, Benjamin/Cummings, Redwood City, CA, pp. 181-209.

Weiss, Mark A. (1996) *Algorithms, Data Structures, and Problem Solving with C++*, Addison-Wesley, Reading, MA, pp. 609-638.

Zipf, G.K. (1935) *The Psycho-Biology of Language: An Introduction to Dynamic Philology*, Houghton Mifflin, Boston, MA, pp. 44-47 (a paperback edition of this book was released by M.I.T. Press in 1965).

Zipf, G.K. (1949) *Human Behavior and the Principle of Least Effort*, Addison-Wesley, Cambridge, MA, pp. 19-55.

Zörnig, Peter and Altmann, Gabriel (1995) “Unified representation of Zipf distributions”,*Computational Statistics & Data Analysis*, vol. 19, pp. 461-473.

Appendix: Programming Assignment

CSC371 Data Structures - Fall 1997
Programming Assignment #2: Zipf’ s Law

Zipf’ s Law is a curious observation about the frequency of words in text. It states that word rank multiplied by word frequency is roughly constant for many text files. Word frequency is the number of times a given word appears in the text file. Words are ranked according to frequency, with the most frequent word given rank 1, the next-most-frequent word given rank 2, etc. Ties are handled by assigning the middle value in a range of ranks. For example, in a two-way tie for most frequent word, each word would be given the rank of 1.5.

Write a C++ program to investigate Zipf’ s Law. Your program should read in words from a text file, counting their frequency of occurrence. Print out the words sorted by frequency (highest frequency first). Within each frequency group, sort the words alphabetically. Then print a table listing word rank, frequency, and rank× frequency to see if Zipf’ s Law holds true.

Test your program on sample text files in `F:\PROGRAMS\DEPT\MA\CSC371` For at least one text file, graph log(rank) vs. log(frequency) using Excel and see if you get a straight line. Turn in this graph with your program.

Implementation Details

Read the input from a text file given as a command-line argument. Program usage is *PA2 textfile*. Print a usage statement if the usage is incorrect, and an error message if the file cannot be opened.

The input text file will consist of a stream of ASCII characters that you must parse into words. Any string of characters separated by white space or punctuation marks is potentially a word. For this assignment, we will restrict words to strings of letters, possibly containing embedded single quotes (contractions). This program will be case-insensitive, so convert all words to lower case. (Hint: The *strtok()* function is very useful in parsing text.)

When a great deal of searching is required, a *hash table* is an appropriate data structure. Store the words and frequencies in a hash table, using a hash function of your choice (make it a good one!). Use linear probing to resolve collisions. Start with a dynamically-allocated hash table of about 1000 entries (remember, the hash table size should be prime). Whenever the hash table gets over 90% full, rehash into a new hash table approximately twice as large as the previous one, up to size 8000 (then you can give up!). Try to do this as efficiently as possible. Print a message to the screen when rehashing occurs.

After hashing all the words in the text file, the hash table will contain at least 10% empty slots. Move all the empty slots to one end of the table so that you can sort the filled slots easily. To sort the table, use the standard library routine *qsort()*. Again, try to make the sorting process as efficient as possible.

Output the program results to two files, with the same name as the input file but different extensions. For example, given an input file named `TEXTFILE.TXT`, print the word concordance to a file named `TEXTFILE.WRD`, and the rank vs. frequency table to a file named `TEXTFILE.ZPF`. Examine the sample output files in the CSC371 directory on FS2 for desirable output formatting. Note that the output files include a header with the input filename, the total number of words, and the total number of distinct words.

Find a partner and work in teams of two on this assignment. Save your program as `PA2.CPP` on your diskette, and submit your work electronically using the SUBMIT program by the due date. Late programs will not be accepted for partial credit. The code should be readable, modular, well-documented, and must run correctly under Turbo C++ 3.0. If your program does not run correctly, indicate why. This will make it easier to give you partial credit.

Five Exercises for Incorporating Moral and Ethical Issues into the Computer Science Curriculum

Dr. Milton C. Wikstrom
Department of Math, Computer Science, and Physics
Wartburg College
Waverly, IA 50677-1003
wikstrom@wartburg.edu

Abstract

As a professor of Computer Science, I find that it is usually a frustrating and difficult task to introduce moral and ethical issues into the computer science curriculum. In this paper, five specific exercises are described that an instructor can use to help students to become more aware of the issues and problems relevant to the field.

Five Exercises for Incorporating Moral and Ethical Issues into the Computer Science Curriculum

Dr. Milton C. Wikstrom
Department of Math, Computer Science, and Physics
Wartburg College
Waverly, IA 50677-1003
wikstrom@wartburg.edu

Abstract

As a professor of Computer Science, I find that it is usually a frustrating and difficult task to introduce moral and ethical issues into the computer science curriculum. In this paper, five specific exercises are described that an instructor can use to help students to become more aware of the issues and problems relevant to the field.

1. Introduction

As part of an upper level, multidisciplinary course that I recently taught (see [Wikstrom 1998]), student groups were ask to present various ethical decision making cases from [Kallman and Grillo 1996]. Each of these cases dealt with situations related to information technology. Not surprisingly, I often found myself at odds with the moral stance that many students had taken. However, this, in and of itself, was not a major concern. I certainly do not consider myself the fountain of all truths. In addition, I do not believe that a professor should force his or her students into holding similar beliefs.

What concerned me, however, was the shallowness of the students' analysis. For example, when considering the morality of copying proprietary software, few students thought past their relative poverty and Bill Gate's wealth. "What about the CS graduate that is looking for work?," I countered pointing out that companies can only hire when they have the means. I continued with the issues of stifled development, overpriced software, the transfer of jobs overseas, an overall diminished economy, and the frustration of having ones own work stolen.

To my further consternation, I soon learned that of the three major liberal arts components (the natural sciences, social sciences, and humanities), it was students in the natural sciences category that exhibited the least depth. Hence, as computer science educators, I feel it behooves us to expose our students to ethical decision making situations whenever we are able. After all, these are the very same students that will someday be writing the software that we will be dependent on.

In this paper, it is my goal to provide educators with five specific exercises that I have found to be successful in the classroom. All five of these exercises are discussed in the appropriate subsections of [2. Five Exercises].

2. Five Exercises

2.1 Simulated Court Trial

The first exercise is to simulate a courtroom scenario with judges (that also serve as the jury), prosecutors, and defendants. The issue at hand is whether or not it was all right for students to make illegal copies of proprietary software. Before starting the exercise, students are required to read relevant articles (for example, see [Mason 1990] and [Nissenbaum 1994]). Next, students are asked to write down three reasons supporting and three reasons opposing the premise. Then students are placed into small groups to compare and discuss their ideas. Once this is accomplished, the class (of about 30 students) is broken in three main groups: a five member panel of neutral judge/jurors, a dozen prosecutors, and a dozen defendants.

To begin the trial, the prosecutors have five minutes to make opening statements. Then the defendants are given five minutes for their position and rebuttal. Next, each side alternates asking the other side for clarification or response to some particular point. At any point in the process, the judges are allowed to step in with questions. In fact, one of the specific duties of the judges is to ask pointed questions whenever a lull occurs. After the issues are hashed over, the judges are asked to confer and to render judgment based solely on the evidence presented.

During the most recent trial, the judges unanimously ruled that the software pirates were guilty as a point of law. However, when the class was surveyed for their actual opinion, a majority (including two of the judges) sided with the defendants. Furthermore, only one person changed their mind on the issue from the opinions held before the exercise. Nevertheless, I still feel that this was a valuable experience for the students since they were required to examine the issues at much greater depth than they had previously done.

2.2 Programming Disasters

The second exercise ultimately focuses on the ethical responsibilities of software developers and their companies to deliver safe and reliable code. To start this exercise, have the students consider the roles of such professional organizations as the American Medical Association or the American Bar Association as they relate to the quality and standards of their respective memberships. Then ask the students to develop a *Code of Ethics* for software developers. Once a reasonable list is developed, introduce the *ACM Code of Ethics and Professional Conduct* [ACM 1992] and the IEEE Code of Ethics [IEEE 1990]. (Incidentally, this is also a good time to bring up your local college's computer usage policies and the students' implicit agreement to abide by them when accepting college entrance.)

In most cases, the students will be surprised and disappointed with the general vagueness and lack of enforcement power that these and other professional organizations promote. However, few if any will be upset about it. Most students are not aware that people have died as a result of sloppy or poorly tested software. At this point, I often regal tales from [Barber 1995], [Bowyer 1996], [Dunlop & Kling 1991], [Schellenberg 1996], and [Schellenberg 1998]. Next I show them the video [WGBH 1992] which graphically illustrates how dangerous computer bugs can be.

2.3 Latch-key Kids Online

The third exercise is non-specific but centers on the easy access to pornography that the internet provides. Without actually demonstrating this, it is very easy to convince students that pornography is rampant on the internet. If you keep an eye on the news, you can usually find a recent story about how a pedophiliac has used a chat room for the purpose of eventually meeting a minor for sex (and sometimes murder). We then discuss the likelihood that a curious latch-key kid will eventually find out about the various free sites.

A spin-off of this subject is censorship and blocking software. One need only point out that no matter what laws are passed, they have no jurisdiction over most off-shore sites.

2.4 Four Step Analysis

[Kallman and Grillo 1996] do a very nice job of introducing and explaining the *Four-Step Analysis Process* to ethical decision making. The fours steps are:

1. understanding the situation
2. isolating the major ethical dilemma
3. analyzing the ethicality of both alternatives from the previous step
4. making a decision and planning the implementation

Step one begins with the listing of all relevant facts. Then each fact is examined to see if it raises an ethical issue. If it does, the question "Why?" is ask and a measure of potential or actual harm is made. Finally, a list of who is involved - the stakeholders - is made.

Step two focuses on isolating the major ethical dilemma by concentrating on what needs to be done now and what should not be done now.

Step three analyzes the situation from the perspectives of: consequentialism, rights and duties, and Kant's categorical imperative. *Consequentialism* involves asking who will be harmed if the action or inaction from step two is followed. Then the student is asked which alternative results in the least amount of harm. It also involves asking who will benefit from action or inaction and who will benefit the most. *Rights and duties* asks what rights have been violated and what duties have been neglected. Then, the particular rights, duties, and stakeholders are identified. *Kant's categorical imperative* asks who will be treated with disrespect or in a different manner than others given that particular actions are or are not taken. Similarly, it asks what benefits will result with the above actions. Then the student is forced to decide which alternatives are preferable under these circumstances.

Step four involves making an actual decision, implementing it, deciding on changes to avoid future entanglements, and identifying past actions that should have been taken that were not.

2.5 Microsoft and the Justice Department

Finally, exercise five involves use of the internet to examine current or recent legal actions related to computers. One particularly good example, that is currently pending in the news, involves the issue of network browsers in the operating system. Microsoft, Inc. and the United States Department of Justice are currently entangled over this issue.

An older issue that is still very relevant dealt with *look and feel* issues. Was Microsoft's Windows software an infringement on Apple's Macintosh interface?

Try relating this issue to copyright laws. Then bring out the opposing perspective by discussing intellectual property laws involving knowledge in someone's head that is planning on changing companies.

3. Final Remarks

While I certainly do not contend that we must impose our own moral views upon our students, I do believe it is incumbent on us to help the students to examine and consider the issues carefully. After all, many of these same students will soon be writing the code that affects our airplanes, nuclear weapons, and virtually every other aspect of our lives.

The number of existing sources speak to these issues. However, within the computer science curriculum, the amount of time expended looking at these same issues is abhorrently slim. The only way to turn the tide, in my opinion, is by having individual professors step forward to start speaking to the issues in every class they teach. With this paper, a few, beginning steps are offered.

References

[ACM 1992] "ACM Code of Ethics and Professional Conduct," Adopted by ACM Council 10/16/92, http://www.acm.org/constitution/code.html

[Barber 1995] Barber, Benjamin R., "Jihad vs. McWorld," *Times Books*, 1995

[Bowyer 1996] Bowyer, Kevin W., "Ethics and Computing, Living Responsibly in a Computerized World," *IEEE Computer Society Press*, 1996

[Dunlop & Kling 1991] "Computerization and Controversy: Value Conflicts and Social Choices," Edited by Dunlop, Charles and Kling, Rob, Academic Press, Inc., Harcourt Brace Jovanovich, Publishers 1991

[IEEE 1990] "Code of Ethics," The Institute of Electrical and Electronics Engineers, Inc., August 1990, see http://www.ieee.org

[Kallman and Grillo 1996] Kallman, Ernest A., & Grillo, John P. (1996). "Ethical Decision Making and Information Technology, An Introduction with Cases," 2nd Edition. McGraw-Hill

[Mason 1990] Mason, Janet, "Warning: Here Come the Software Police," originally appearing in *Across the Board*, October 1990, reprinted in *Computer Studies: Computers in Society*, Sixth Edition, Edited by Kathryn Schellenberg, Dushkin Publishing Group/Brown & Benchmark Publishers, 1996

[Nissenbaum 1994] Nissenbaum, Helen, "Should I Copy My Neighbor's Software?," originally appearing in *Computers, Ethics, & Social Values*, Prentice Hall, 1994, reprinted in *Computer Studies: Computers in Society*, Sixth Edition, Edited by Kathryn Schellenberg, Dushkin Publishing Group/Brown & Benchmark Publishers, 1996

[Schellenberg 1996] "Computer Studies: Computers in Society", 6th Edition, edited by Kathryn Schellenberg, Duskin Publishing 1996

[Schellenberg 1998] "Computer Studies: Computers in Society", 7th Edition, edited by Kathryn Schellenberg, Duskin Publishing 1998

[WGBH 1992] "The Machine That Changed the World; Part 5 - The World At Your Fingertips," (video), WGBH (Public Television Station in Boston), Princeton, N.J.: Films for the Humanities and Sciences 1992

[Wikstrom 1998] Wikstrom, Milton C., "The Global Information Society: A Multidisiplinary Course," *The 31st Annual Small College Computing Symposium*, North Dakota State University, Fargo, N.D., April 1998

The Global Information Society: A Multidisciplinary Course

Dr. Milton C. Wikstrom
Department of Math, Computer Science, and Physics
Wartburg College
Waverly, IA 50677-1003
wikstrom@wartburg.edu

Abstract

The newly emerging information age is having a dramatic impact on individuals, societies, and governments. This paper describes a new multidisciplinary course that examines a variety of issues related to technology and information. Course topics include technological impact on the arts to the changing workplace to global power struggles. Grading criteria, hints, and internet sources are discussed.

The Global Information Society: A Multidisciplinary Course

Dr. Milton C. Wikstrom
Department of Math, Computer Science, and Physics
Wartburg College
Waverly, IA 50677-1003
wikstrom@wartburg.edu

Abstract

The newly emerging information age is having a dramatic impact on individuals, societies, and governments. This paper describes a new multidisciplinary course that examines a variety of issues related to technology and information. Course topics include technological impact on the arts to the changing workplace to global power struggles. Grading criteria, hints, and internet sources are discussed.

1. Introduction

In 1997, a new class was offered at Wartburg College that explored the effects of technology and the information age on people. The catalog description [Wartburg, 1996] for this course appears in [Fig. 1]:

> ID320 - THE GLOBAL INFORMATION SOCIETY
>
> The newly emerging information age is having a dramatic impact on individuals, societies, and governments. This course examines a variety of issues ranging from the impact on the arts to the changing workplace to global power struggles.

Figure 1 : Catalog Description

The *ID* in ID320 stands for *interdisciplinary*. This course was offered for the first time during 1997 and was well received by students and faculty. The only prerequisite is that the student be of third-year standing or higher. It is also designated as a “writing intensive” course which means that a great deal of attention is paid to both the writing process as well as the final written product. All students are required to take one ID and two writing intensive courses in order to graduate. Course enrollment is limited to 24 students.

The primary goal of the course in 1997 was to obtain a non-trivial awareness and understanding of the issues and questions listed below:

1. What is the Information Age and how does it differ from other ages?
2. How will the Global Information Society (GIS) impact a typical U.S. citizen?
3. How will the GIS impact citizens of third world nations?
4. What are the respective agendas in the American, European, and Far East spheres of influence for seizing a global advantage?
5. In what way are various areas of discipline being affected by technology and the GIS?

6. What are the effects of GIS on the workplace?
7. What are the effects of GIS on the family?
8. What are the dangers inherent in GIS? Can we reduce their impact?
9. In what way are regional, national, and international markets and economies driven or affected by GIS?
10. How will GIS help or hinder the environment.

As a multidisciplinary course, a heavy emphasis was required in two of the three liberal arts areas. In addition, it must also speak to the third area - albeit to a lesser extent. These three areas are: the natural sciences, the humanities, and the social sciences.

From an instructor's perspective, teaching such a course is problematic. No one person has expertise in all of these areas. The solution, I found, was to rely heavily on both the knowledge of the students and on the internet as a source of timely and relevant information. A text was also utilized [Kallman and Grillo 1996] that focused on ethics case studies related to information technology.

In [2. Course Description], I provide a much more thorough description of the course. Naturally, such a course would be laughable without heavy reliance on the internet. See [3. The Internet As A Text] for details about internet sources and the role of the internet in this course. In [4. Writing Intensive], discussion includes ways to incorporate a heavy writing emphasis to such a course. Included in [4. Writing Intensive] are samples of the grading criteria used by students and the instructor. Next, included in [5. Final Remarks] is a brief discussion about the successes and failures of this first time course offering. Finally, some suggestions are proffered for offering a similar class in the future.

2. Course Description

2.1 General Information

At my particular institution, we operate under a 4-4-1 semester basis. This means that students take four course credits in the fall, four in the winter, and one during May term. This class was taught during the May term. As a result, students had three hours of "seat time" every day for 19 class days. During each of the three hour blocks of time, there was a combination of some or all of the following: lecture, group work, laboratory, class discussion, ethics case study analyses, guest speakers, and student presentations. Students were expected to perform an additional four to six hours of work outside of class time per day.

2.2 Course Syllabus

2.2.1 Sources

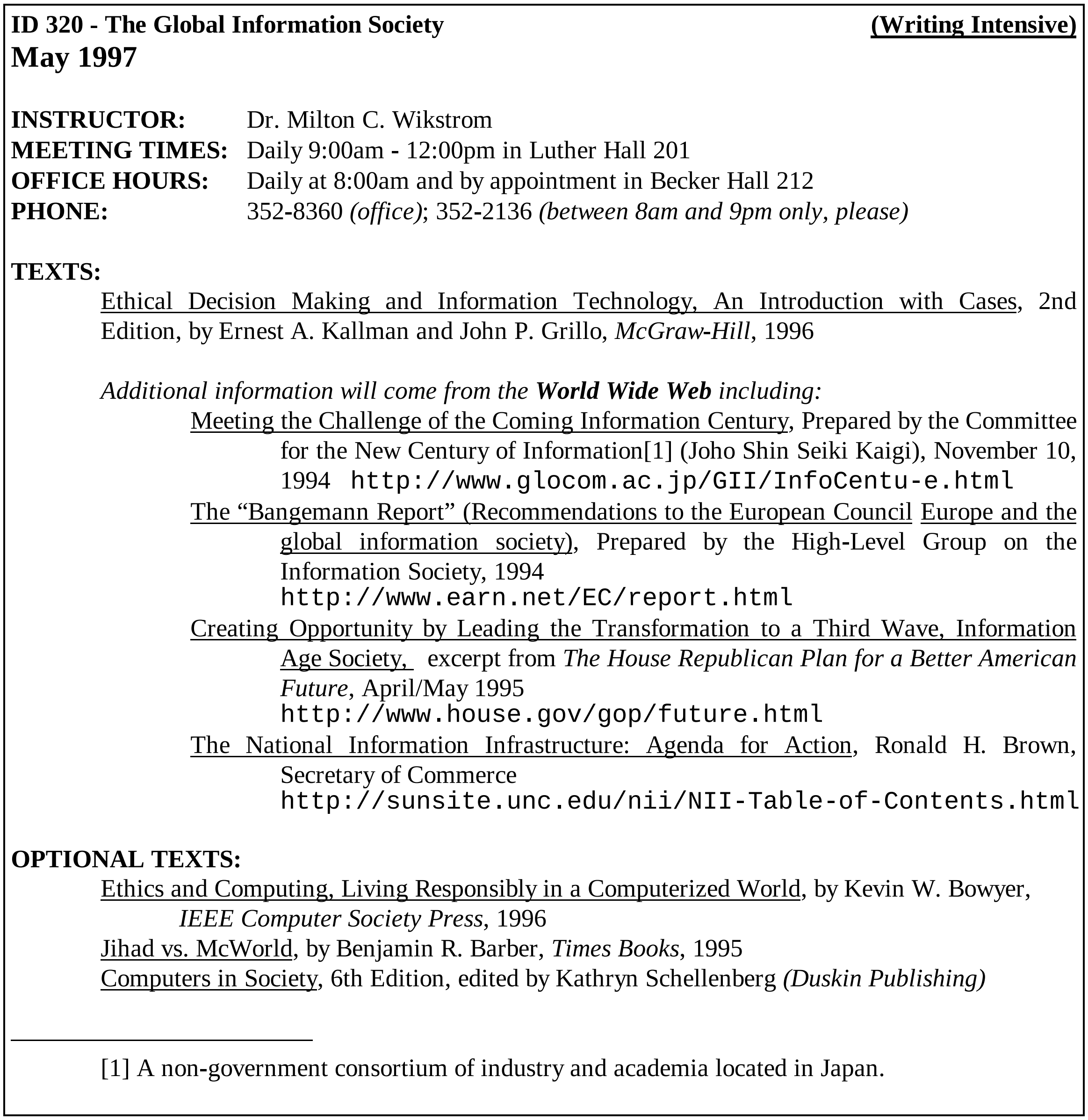
The portion of the course syllabus is shown in [Fig. 2] with additional relevant portions shown as running text below. As [Fig. 2] illustrates, there was one required text: [Kallman and Grillo 1996]; three optional texts: [Bowyer 1996], [Barber 1995], and [Schellenberg 1996]; and four specified web sites: [Kaigi 1994], [Bangemann 1994], [Republican Plan 1995], and [Brown 1995]. Notice in [References] that the web site addresses have changed since the creation of the syllabus.

2.2.2 Goals

The primary goal of this course is to obtain a non-trivial awareness and understanding of the issues and questions listed below:

1. What is the Information Age and how does it differ from other ages?
2. How will the Global Information Society (GIS) impact a typical U.S. citizen?
3. How will the GIS impact citizens of third world nations?
4. What are the respective agendas in the American, European, and Far East spheres of influence for seizing a global advantage?
5. In what way are various areas of discipline being affected by technology and the GIS?
6. What are the effects of GIS on the workplace and the family?
7. What are the dangers inherent in GIS? Can we reduce their impact?
8. In what way are regional, national, and international markets and economies driven or affected by GIS?
9. How will GIS help or hinder the environment?

Secondary goals of this course include: learning to effectively use the World Wide Web, to improve verbal and written communication skills, and to integrate a diverse number of disciplinary topics into a single and more mature world view.



ID 320 - The Global Information Society **(Writing Intensive)**

May 1997

INSTRUCTOR: Dr. Milton C. Wikstrom
MEETING TIMES: Daily 9:00am - 12:00pm in Luther Hall 201
OFFICE HOURS: Daily at 8:00am and by appointment in Becker Hall 212
PHONE: 352-8360 *(office)*; 352-2136 *(between 8am and 9pm only, please)*

TEXTS:

Ethical Decision Making and Information Technology, An Introduction with Cases, 2nd Edition, by Ernest A. Kallman and John P. Grillo, *McGraw-Hill*, 1996

Additional information will come from the ***World Wide Web*** *including:*

Meeting the Challenge of the Coming Information Century, Prepared by the Committee for the New Century of Information[1] (Joho Shin Seiki Kaigi), November 10, 1994 `http://www.glocom.ac.jp/GII/InfoCentu-e.html`

The "Bangemann Report" (Recommendations to the European Council Europe and the global information society), Prepared by the High-Level Group on the Information Society, 1994
`http://www.earn.net/EC/report.html`

Creating Opportunity by Leading the Transformation to a Third Wave, Information Age Society, excerpt from *The House Republican Plan for a Better American Future*, April/May 1995
`http://www.house.gov/gop/future.html`

The National Information Infrastructure: Agenda for Action, Ronald H. Brown, Secretary of Commerce
`http://sunsite.unc.edu/nii/NII-Table-of-Contents.html`

OPTIONAL TEXTS:

Ethics and Computing, Living Responsibly in a Computerized World, by Kevin W. Bowyer, *IEEE Computer Society Press*, 1996
Jihad vs. McWorld, by Benjamin R. Barber, *Times Books*, 1995
Computers in Society, 6th Edition, edited by Kathryn Schellenberg *(Duskin Publishing)*

[1] A non-government consortium of industry and academia located in Japan.

Figure 2 : Syllabus, Page 1

2.2.3 Projects, Papers, & Presentations

As this is a writing intensive course, all students are expected to develop three papers. The first paper is to be 3 to 4 pages in length and is to directly address one of the issues listed above. The second paper is to explicitly address the impact of GIS on the third world nations. This paper may be done individually (3 to 4 pages) or collaboratively in groups of at most 4 people (add one additional page in length per additional group member). Finally, the last paper is to discuss the affects of GIS on a student's own chosen area of study. This paper is to be 5 to 6 pages in length. This paper may also be done in groups providing that they consist of none of the same partners from the previous paper. In addition to writing the third paper, the individual or group must also present the contents of the paper or a related project to the class. Students electing to do a related project can earn up to 10 extra percentage points to be applied to their final grades. All paper topic and project proposals must have the instructor's pre-approval before the paper is written or the project is actually implemented. In addition, all students are expected to help critique the various papers and presentations. All papers must be written using a word processor!

2.2.4 Grading

[Fig. 3] contains the relative grading weights used in the course. Notice that the third paper presentation was weighted at 15%. The idea was to treat this task as the class final exam. It was also intended to stress the importance and seriousness of the task. A similar tact was taken for critiquing each others rough drafts. Students were expected to take this task very seriously as well. Once the constructive criticism was delivered and the students were given a few days to update their papers, they were then asked to give the reviewer a grade based on how helpful and in depth they were in the original review. Finally, notice that class participation was weighted at 15%. This was strong encouragement to keep a constant dialogue going in the class and to ensure strong participation during ethics case study sessions.

2.3 Final Course Remarks

In all, I probably lectured about ten percent of the entire term. One of the more difficult but ultimately satisfying challenges of this course was to, indeed, reach a *nontrivial* understanding on as many of the goal issues as possible. Another goal of the course was to improve students' communication skills. More detailed information on this topic appears in [4. Writing Intensive].

3. The Internet As A Text

3.1 Getting Started

GRADING:

Students will complete two shorter and one longer paper. In addition, a presentation over the last paper or some related project is expected. Grades will also be based on class participation, paper and presentation critiques, and for any quizzes and exercises that are assigned. Your final grade will be determined by straight 90 / 80 / 70 / 60 percentage (*i.e.* no class curve). For every two inexcused absences, your grade will drop one full letter (e.g. 2 or 3 inexcusable absences would drop a B+ to a C+). Grades will be calculated using a weighted average as follows:

2 Short Papers (@ 15% each)	30%
1 Longer Paper	20%
Critiques	10%
Quizzes/Homework/Presentation	10%
Paper #3 Presentation	15%
Class Participation	15%
	(Total 100%)

Figure 3 : Syllabus - Grading

In my personal opinion, the heavy use of the internet as a primary source is the single most important factor contributing to the success of this course. First, a few months to a year before the course starts, the instructor should use his or her favorite search engine to search for relevant sources related to the topic of the *global information society*. As a result of these searches, several major themes and questions will likely emerge. It is around these themes that the overall course content and schedule can be developed. I do not believe the same set of questions should be reused in later semesters since many will go out of date quickly or be surpassed by more timely ones.

In my particular case, the theme of how world powers were struggling to deal with the reality of the global information society emerged as the major theme. This is why I placed the four specified web sites: [Kaigi 1994], [Bangemann 1994], [Republican Plan 1995], and [Brown 1995], directly into the syllabus.

However, one must be careful to not become too dependent on any set of web sources. Due to their transient nature, they have a nasty habit of moving or disappearing without warning. For example, three of the four above sites had moved in between the time that I printed my syllabus and the point in class that the sites were needed. Fortunately, I was able to find them at new locations. (The *current* addresses, as this paper is being written, for [Kaigi 1994], [Bangemann 1994], and [Brown 1995], are given in [References]. I was not able to locate a current address for [Republican Plan 1995].) One moral of the story is to download any crucial web-sourced information.

3.2 Student Use of the Web

In a course such as this, it is not uncommon to have several students with limited or no prior knowledge of computers and web browsers. Hence, it is important to take them into a computer laboratory at the start of the course to familiarize them with these tools. A series of initial assignments that force the students to find information about current technology related events, about their majors, or about virtually any specific topic is most helpful. Encourage students to use the web as a source for their paper assignments. I found that students quickly adapted to (and very much enjoyed) using the web in this way. One particularly nice consequence of this was that class discussions often included unexpected and very fresh topics that had never occurred to the instructor.

One particularly pleasant surprise associated with the web occurred when students were researching the particular third-world nation that had been assigned to them. Through various web sites, many student groups were able to obtain the telephone number for the embassy of their assigned country. When telephoned, without exception, each embassy went out of their way to provide information about technological advances in their countries. Also, due to the staff sizes in these embassies, at least one group spoke directly to the ambassador himself. The above situation also provided a nice example to explain and discuss the accuracy (or inaccuracy) of information on the web.

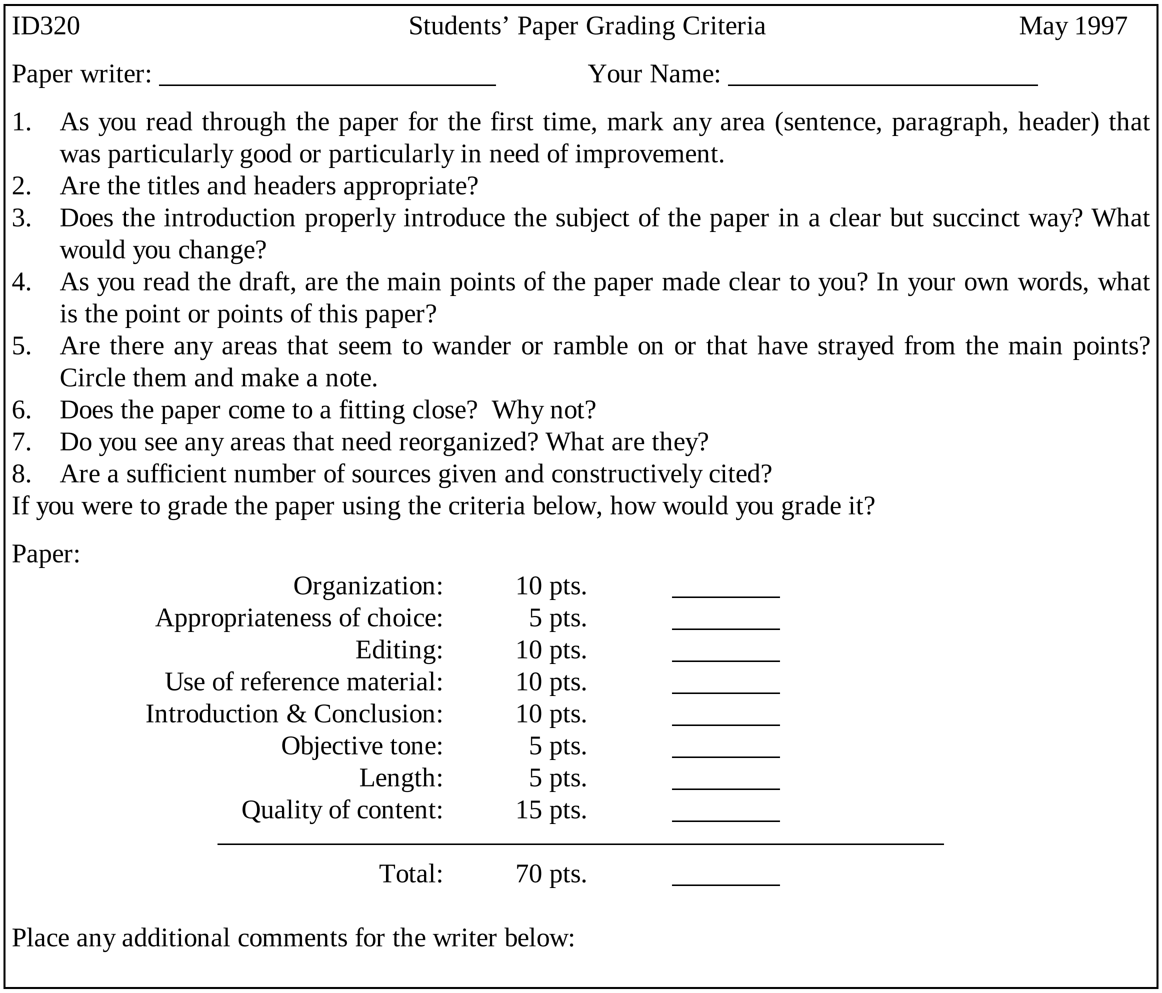
3.3 Create a Homepage

Another good idea is to place your course information, students papers, and interesting sites into the classes' own homepage. Since I did not have the know-how at the time, I ask some of the more technologically proficient students to do this as an alternative to the final paper. Of course, one should be careful to check with the class in advance since it is their work which will be on display. In the end, I believe the quality of the papers was improved by having taken this approach. Unfortunately, time and various other reasons got in the way of actually putting this site live, on-line.

4. Writing Intensive

Towards the goal of improving students' communication skills, there were assigned: three major papers, a major presentation, and the leading of a class discussion regarding an ethics related case study. Besides writing the papers, the students were very much involved with process. In addition to meetings with the instructor, outlines and rough drafts were required and there were student critiques of other students' rough drafts and critiques of reviews. The forms used by students for rough draft evaluation, by the instructor for the paper evaluation, and by students for evaluation of reviewers are shown in [Fig. 4], [Fig. 5], and [Fig. 6], respectively.

The topic of the first paper was left open to the students providing that it addressed one of the questions listed in [2.2.2 Goals]. The second paper was to focus on the impact of technology on third world nations. As a result of group work successes in other classes (see [Wikstrom 1997]), both the second and third papers could be written by groups. Finally, the third paper was to focus on how technology was impacting the students' chosen area of study. To facilitate group formation for this paper, students were first divided into groups based on their majors. Fortunately, the number of majors represented was large but not so large that many where in groups of one. In such cases, the instructor found common interests that the students could focus on. A presentation was also required from each third paper group, on the last day of class, related to their third paper topic.



ID320 Students' Paper Grading Criteria May 1997

Paper writer: _________________________ Your Name: _______________________

1. As you read through the paper for the first time, mark any area (sentence, paragraph, header) that was particularly good or particularly in need of improvement.
2. Are the titles and headers appropriate?
3. Does the introduction properly introduce the subject of the paper in a clear but succinct way? What would you change?
4. As you read the draft, are the main points of the paper made clear to you? In your own words, what is the point or points of this paper?
5. Are there any areas that seem to wander or ramble on or that have strayed from the main points? Circle them and make a note.
6. Does the paper come to a fitting close? Why not?
7. Do you see any areas that need reorganized? What are they?
8. Are a sufficient number of sources given and constructively cited?

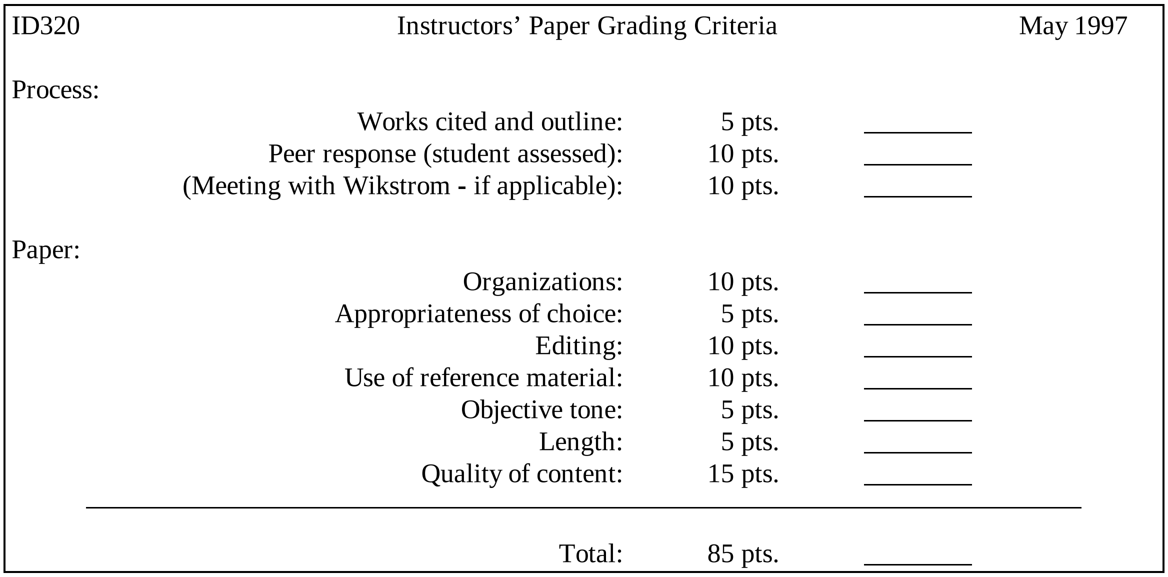
If you were to grade the paper using the criteria below, how would you grade it?

Paper:

Organization:	10 pts.	________
Appropriateness of choice:	5 pts.	________
Editing:	10 pts.	________
Use of reference material:	10 pts.	________
Introduction & Conclusion:	10 pts.	________
Objective tone:	5 pts.	________
Length:	5 pts.	________
Quality of content:	15 pts.	________
Total:	70 pts.	________

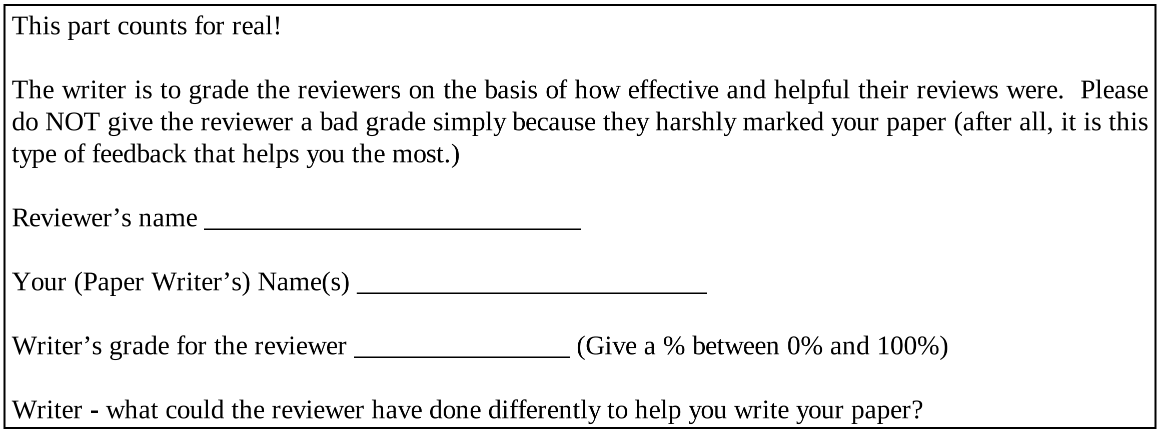
Place any additional comments for the writer below:

Figure 4 : Students' Rough Draft Grading Criteria



ID320	Instructors’ Paper Grading Criteria		May 1997
Process:			
	Works cited and outline:	5 pts.	________
	Peer response (student assessed):	10 pts.	________
	(Meeting with Wikstrom - if applicable):	10 pts.	________
Paper:			
	Organizations:	10 pts.	________
	Appropriateness of choice:	5 pts.	________
	Editing:	10 pts.	________
	Use of reference material:	10 pts.	________
	Objective tone:	5 pts.	________
	Length:	5 pts.	________
	Quality of content:	15 pts.	________
	Total:	85 pts.	________

Figure 5 : Instructor's Paper Grading Criteria



This part counts for real!

The writer is to grade the reviewers on the basis of how effective and helpful their reviews were. Please do NOT give the reviewer a bad grade simply because they harshly marked your paper (after all, it is this type of feedback that helps you the most.)

Reviewer’s name ____________________________

Your (Paper Writer’s) Name(s) __________________________

Writer’s grade for the reviewer ________________ (Give a % between 0% and 100%)

Writer - what could the reviewer have done differently to help you write your paper?

Figure 6 : Students' Grading of Reviewers

5. Final Remarks

The name of the game is *flexibility*. If you are a person that delights in a learner-centered classroom with constantly changing themes and sources, then this may be the type of class that you would enjoy teaching. Be forewarned, however, that success also depends a great deal on the student demographics. In my particular classroom, I had a wide range of majors represented. There was also several international and minority students. This helped even more to enhance the variety of good sources, diverse perspectives, and stimulating discussion ideas.

Some lessons learned include:

- start early on creating your own web site
- don’t become too reliant on particular web sources

- download web sourced information if it is crucial
- rely on your students' knowledge and background - especially concerning their majors
- do not assume the students hold the same set of moral standards that you do - extensive use of ethics case studies merges quite nicely with many of the relevant class issues
- have fun

In the end, I really did have fun teaching this course and I learned a lot in the process. Based on the end-of-term *Student Ratings of Instruction* form results, the students did as well.

References

[Bangemann 1994] (The *Bangemann Report*) "Recommendations to the European Council Europe and the Global Information Society," Prepared by the High-Level Group on the Information Society, 1994 http://www.clark.net/pub/davidcol/bangeman.htm

[Barber 1995] Barber, Benjamin R., "Jihad vs. McWorld," *Times Books*, 1995

[Bowyer 1996] Bowyer, Kevin W., "Ethics and Computing, Living Responsibly in a Computerized World," *IEEE Computer Society Press*, 1996

[Brown 1995] Brown, Ronald H., Secretary of Commerce, "The National Information Infrastructure: Agenda for Action," http://sunsite.unc.edu/nii/NII-Table-of-Contents.html

[Kaigi 1994] Meeting the Challenge of the Coming Information Century, Prepared by the Committee for the New Century of Information (Joho Shin Seiki Kaigi), November 10, 1994 http://www.glocom.ac.jp/gii/infocentu%2De.html

[Kallman and Grillo 1996] Kallman, Ernest A., & Grillo, John P. (1996). "Ethical Decision Making and Information Technology, An Introduction with Cases," 2nd Edition. McGraw-Hill

[Republican Plan 1995] "Creating Opportunity by Leading the Transformation to a Third Wave, Information Age Society," excerpt from "The House Republican Plan for a Better American Future," April / May 1995 http://www.house.gov/gop/future.html

[Schellenberg 1996] "Computer Studies: Computers in Society", 6th Edition, edited by Kathryn Schellenberg, Duskin Publishing 1996

[Wartburg 1996] Wartburg College 1996-98 Academic Catalog, Volume LVII (August 1996). Course Descriptions. Waverly, IA: Wartburg College

[Wikstrom 1997] Wikstrom, Milton C., "Group Semester Projects in CS1 and CS2," *The 30th Annual Small College Computing Symposium*, University of Wisconsin - Parkside, April 1997