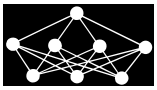

# AN ALGORITHM FOR TEST DATA SET REDUCTION FOR WEB APPLICATION TESTING

*A. Askarunisa*[*], *N. Ramaraj*[†]

**Abstract:** Web Applications have become a critical component of the global information infrastructure, and it is important that they be validated to ensure their reliability. Exploiting user session data is a promising approach to testing Web applications. However, the effectiveness of user session testing technique depends on the set of collected user session data: The wider this set, the greater the capability of the approach to detect failures, but the wider the user session data set, the greater the cost of collecting, analyzing and storing data. In this paper, a technique for reducing a set of user sessions to an equivalent smaller one is implemented. This technique allows reducing of a wider set of user sessions to an equivalent reduced user session and pages, sufficient to test a Web application effectively. Reduction of a user session for several web applications like TCENet Web application, Portal application, Social Networking, Online shopping, Online Library is carried out in order to validate the proposed technique; and our technique is compared with HGS, Random Reduction technique and the Concept Lattice technique to evaluate its efficiency.

Key words: *User session, test case reduction, row dominance, column dominance, testing effectiveness, fault detection density, fault detection efficiency*

*Received: April 19, 2010*
*Revised and accepted: December 15, 2010*

## 1. Introduction

User-session data based techniques have been proposed in the literature as a way for effectively testing an existing Web Application (WA). These capture/replay techniques collect data about user interactions with the Web server and transform them into test cases using a particular strategy.

[*]A. Askarunisa
Department of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai, E-mail: nishanazer@yahoo.com
[†]N. Ramaraj
GKM College of Engineering, Anna University, Chennai, E-mail: aacse@tce.edu

 

One of the main advantages of this approach is the possibility of generating test cases without analyzing the internal structure of the web application, thus reducing the costs of finding inputs. In addition, generating test cases using user session data is less dependent on the heterogeneous and fast changing technologies used by Web Applications (WAs), which is one of the major limitations of white box testing techniques.

However, the effectiveness of user-session based testing techniques depends on the set of user session data collected: The wider this set, the greater the effectiveness of the approach to detect failures, but the wider the user session data set, the greater the cost of collecting, analyzing and storing data. Therefore, there is a trade-off between test suite size and failures detection capability.

A specific concern of user-session based testing consists of designing test suites that are able to cover each possible user behavior, including both frequent and less frequent ones. Generally, user sessions may contain more replicas of frequent behaviors. Therefore, to design a test suite effectively from a given set of user sessions, it may be useful to group user equivalent behaviors together into equivalence classes, and to define test cases for covering each equivalence class.

This classification problem can be solved by analyzing the classes of output data produced during the navigation. Equivalent output classes are, indeed, indicators of equivalent user behaviors, so if we are able to classify output data, we will find classes of equivalent behaviors too. The problem can be approached differently, depending on the involved category of Web applications. As an example, for dynamic Web applications composed of client pages (deployed at the client side of the application) and server pages (executed at the server side of the application and including script modules), Server pages are mainly responsible for the business logic and for producing output to user requests. Server pages output can be returned as static client pages (i.e. pages stored in a static html file) or as Built Client Pages (BCPs), i.e. pages dynamically built at run time by server pages, whose content depends both on user input and on the state of the application data. A server page is able to build one or more types of client pages, where each type of page can be associated with a logically different output, corresponding to a different application behavior/scenario.

As an example, a server page that processes user identification data will build either an access denied page (in case of invalid input) or a successful access page allowing the user to proceed with the elaboration. These two output pages are alternatively produced and represent the output of two distinct scenarios of the use case 'User Validation'.

As a consequence, the problem of identifying and classifying equivalent behaviors contained in a set of user sessions can be transformed into the problem of identifying and classifying the built client pages into equivalence classes. This problem has already been addressed in the literature, where different techniques have been proposed to identify groups of equivalent BCPs [5, 6] for the aim of reverse engineering an existing Web application.

In this paper, a technique for reducing a set of user sessions to an equivalent smaller one is implemented. This technique, namely the ERC (Essential Row dominance and Column dominance) reduction technique, allows equivalent behaviors included in user sessions to be identified and grouped together into equivalence

classes, and produces a reduced set of user sessions that can be used to design test suites with a reduced effort. Some preliminary case studies were carried out to validate the proposed technique and to evaluate its effectiveness. The results of a case study will be presented in the paper. As to the organization of the paper, Section 2 describes related work on Web application testing, while Section 3 presents the ERC reduction technique. In Section 4, the results of the case study we carried out for evaluating this approach will be provided, while Section 5 discusses conclusive remarks and future work.

## 2. Related Work

Various user-session based WA testing techniques exploit captured data about user interactions with the web server, such as the clients' requests expressed in form of URLs and name-value pairs. These data can be obtained from the files stored by web servers, or by instrumenting Web pages for capturing the requested page and name-value pairs of exchanged parameters. The captured data about user sessions can be transformed into a set of HTTP requests, each one providing a separate test case.

Elbaum et al. [5, 6] proposed a user session approach to test a Web application. In his study, collected user session data consists of sequences of http requests made by users. Each sequence reports the pages (both client and server ones) the user visited, together with the data he/she considered as input, as well as the data resulting from the elaboration of requests the user made.

Sampath et al. [4] have explored the possibility of using a concept analysis for achieving scalability in user-session based testing of Web applications. The concept analysis is a technique for clustering objects that have common discrete attributes; It is used to reduce a set of user sessions to a minimal test suite that still represents the actual executed user behavior.

HGS algorithm named after Harrold, Guptha and Soffa [8] is used for reducing the number of user sessions. It selects a representative set from the original by approximating the optimal reduced set, it uses the requirement cardinality which should be the # of test cases covering that requirement and it selects the most frequently occurring test case with the lowest requirement cardinality.

The random reduction algorithm is also used for reducing the number of user sessions. This algorithm randomly selects some user sessions and removes them.

Sampath et al. [4] have proposed a test suite reduction technique that assures the coverage of all the URLs contained in the set of user sessions; our user session reduction technique will assure the coverage of both the URLs and the classes of equivalent built client pages included in the set of user sessions. Therefore, our approach will provide a different coverage criterion.

## 3. Problem Description

Test suite sizes may grow significantly with modifications to the software over time. Due to time and resource constraints, test suite minimization techniques attempt to remove those test cases from the test suite that have become redundant over

time since the requirements covered by them are also covered by other test cases in the test suite. Earlier work has shown that test suite minimization techniques can severely compromise the fault detection effectiveness of test suites.

The main goal of the paper is to

1. Improve the fault detection effectiveness of reduced suites without severely affecting the extent of test suite size reduction.
2. To selectively retain the redundant test cases in a reduced suite, which may be redundant with some testing criteria but not redundant according to other testing criteria.

In this paper, a technique for reducing a set of user sessions to an equivalent smaller one is implemented. This technique, namely the ERC reduction technique, allows equivalent behaviors included in user sessions to be identified and grouped together into equivalence classes, and produces a reduced set of user sessions that can be used to design test suites with a reduced effort. Some preliminary case studies were carried out to validate the proposed technique and to evaluate its effectiveness.

## 3.1 The Essentiality Row dominance Column Dominance (ERC) reduction technique

We have implemented a reduction technique called the Essentiality Row dominance Column dominance (ERC) technique. This has been defined for supporting the efficient generation of test suites for Web applications by exploiting a set of user session data. This technique has been developed for testing dynamic Web applications that use script modules on the server side for generating HTML client pages at run time. This technique aims at classifying equivalent interaction scenarios exercised by a set of user sessions and reducing the original set of user sessions into an equivalent smaller one.

A user session is composed of a sequence of http requests made by users, where each request is made up of a pair comprehending the requested Web page and the exchanged data. Each Web page may be either a client page (deployed at the client side of the application) or a server page (executed at the server side of the application). Client pages may be either static or dynamic. A server page is able to build a finite set of types of client pages, and each type of built client page defines an Equivalence Class of Built Client Page (EBCP).

Fig. 1 gives a detailed framework of the reduction process using the ERC technique. The access log of the Web application under test is collected from the log parser and the user sessions instrumented. The ERC reduction algorithm is applied by calculating the row dominance and column dominance. The reduction strategy can be used in the context of a test suite generation process that will include the following steps:

1. User session collection and instrumentation
2. Reduction of the set of collected user sessions
3. Test suite generation

These steps are explained as follows.

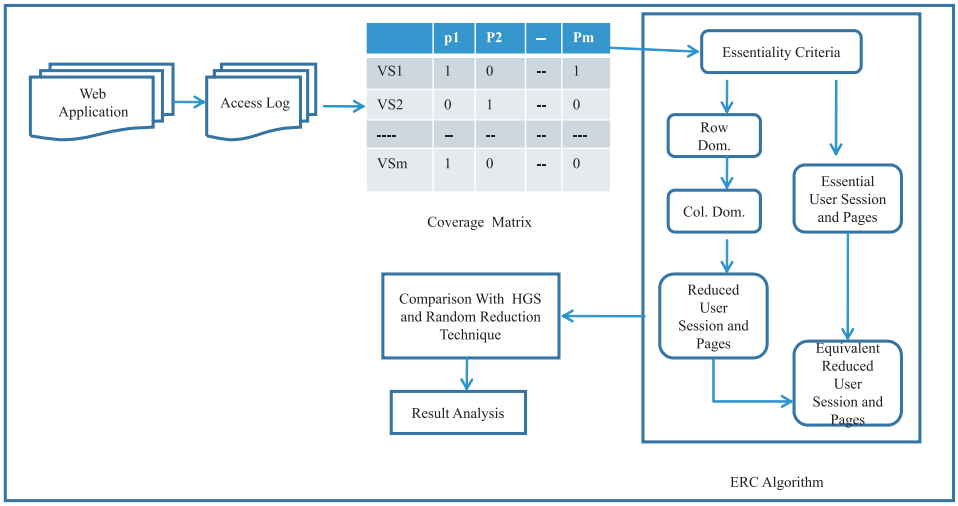


**Fig. 1** *Framework of the ERC reduction technique.*

## 3.2 User session collection and instrumentation

In the first phase of the process, a set of user sessions as in Figure 2 was collected.

These user sessions are collected for the Bookstore application from GOTO Code.com [4]. The sessions are then instrumented for generating a set of sessions where each session is composed of a sequence of http requests made by users. Each sequence will report the pages (both client and server ones) the user visited together with the data he/she inputted, as well as the built client pages resulting from the elaboration of requests the user made.

Collected user sessions will usually exercise different use case interaction scenarios. However, each scenario produces a logically different output that is provided by a different type of built client page in a dynamic Web application.

The presence of built pages in a user session, each one corresponding to a given EBCP, will indicate which interaction scenarios are covered by that user session. The coverage relationship between EBCPs and user sessions can be represented by a binary matrix $M$, called *Coverage Matrix*, where each row of $M$ corresponds to a user session and each column corresponds to an EBCP:

$$m(i,j) = 1 \textit{ iff user session } i \textit{ contains a page of the EBCP } j$$

$$m(i,j) = 0 \textit{ iff user session } i \textit{ does not contain a page of the EBCP } j.$$

The user session is instrumented manually, thereby converting the user session dataset into coverage matrix $M$ consisting of the user session and the Equivalent Built Client Pages (EBCPs).


**Fig. 2** *User session log.*

### 3.3 Reduction of the collected user sessions

A set $U$ of user sessions that cover a given set of EBCPs can be transformed into an equivalent reduced set containing the minimum number of user sessions exercising the same set of EBCP$s$.

The reduction technique exploits an algorithm that is usually adopted for the synthesis of combinatorial machines. This algorithm can be applied to the coverage matrix $M$ and it uses the following essentiality and dominance criteria for generating the reduced set of user sessions:

- ***Essentiality criterion***:

  A user session US is called essential if it is the only user session from $U$ containing a page of a given EBCP;

- ***Row dominance criterion***

  A user session $\mathrm{US}_i$ dominates $\mathrm{US}_j$ if all EBCPs contained in $\mathrm{US}_j$ are also in $\mathrm{US}_i$;

- ***Column dominance criterion***

  An EBCP $\mathrm{P}_{i(I-index)}$ dominates an EBCP $\mathrm{P}_{j(J-index)}$ if $\mathrm{P}_{i(I-index)}$ is contained in all user sessions containing $\mathrm{P}_{j(J-index)}$

The first criterion defines essential user sessions that will be included in the final reduced user session set and allows the coverage matrix M to be simplified by deleting its essential rows and the set of columns in the essential user sessions.

The last two criteria allow the matrix to be simplified by deleting its dominated rows and dominant columns. The Web Application (WA) to be tested is chosen initially and the number of pages in the WA is identified. The access log containing the individual behaviors of users is collected. With the contents of access log, the pages accessed by the users are identified manually, and thus the coverage matrix is obtained. A sample access log is shown in Fig. 3.


**Fig. 3** *Sample access log.*

Let us consider a simple example to explain how the previous criteria can be used to solve the reduction problem. Let's consider the coverage matrix M reported in Tab. I: It comprehends 5 user sessions (from US1 to US5) and 5 EBCPs (from P1 to P5).

The reduction comprehends the following steps:

1. Since P2 is an essential page, the corresponding user session US2 and the associated EBCPs with US2 are deleted (as in Tab. II), which will be included in the final reduced matrix.

| | P1 | P2 | P3 | P4 | P5 |
|---|---|---|---|---|---|
| US1 | 1 | 0 | 0 | 1 | 0 |
| US2 | 0 | 1 | 0 | 1 | 1 |
| US3 | 1 | 0 | 1 | 0 | 0 |
| US4 | 1 | 0 | 0 | 0 | 1 |
| US5 | 0 | 0 | 1 | 1 | 1 |

**Tab. I** *Initial coverage matrix.*

| | P1 | P3 |
|---|---|---|
| US1 | 1 | 0 |
| US3 | 1 | 1 |
| US4 | 1 | 0 |
| US5 | 0 | 1 |

**Tab. II** *Coverage matrix after essentiality criteria.*

2. US3 is dominating US1, US4, US5 since US1, US4, US5 are contained in U3, hence the dominated rows are deleted (as in Tab. III).

| | P1 | P3 |
|---|---|---|
| US3 | 1 | 1 |

**Tab. III** *Coverage matrix after row dominance criteria.*

3. P1 and P3 dominates each other since P1 is contained in P3 and vice-versa, hence the dominant column is deleted (consider P3 in this case, as in Tab. IV).

| | P3 |
|---|---|
| US3 | 1 |

**Tab. IV** *Coverage matrix after column dominance criteria.*

| | P2 | P3 | P4 | P5 |
|---|---|---|---|---|
| US2 | 1 | 0 | 1 | 1 |
| US3 | 0 | 1 | 0 | 0 |

**Tab. V** *Coverage matrix after adding essential user sessions and pages.*

4. The final reduced user session including the essential user session US2 and the associated EBCPs P2, P4, P5 is given in Tab. V.

This technique assures that the reduced set will contain the minimum number of user sessions exercising the same set of EBCP*s* from the original set, without

considering the URLs ordering in the user sessions. Therefore, this method will be useful in a testing process that just aims at covering the single built pages, independently of their preconditions.

### 3.4 Test suite generation

This third step aims at producing a test suite from user session collected data. Given the reduced set of user sessions, the test suite can be generated with different strategies. As an example, the simplest approach may consist of associating each user session with a distinct test case. Another approach may require that data from distinct user sessions are intertwined in order to obtain further test cases.

## 4. Methodology

In order to validate the ERC reduction technique, some preliminary case studies were carried out. In this section, the results of a case study involving a real Web application will be presented.

Our Web application is TCENet (`www.tce.edu`). It is a student portal website. Academic details of every student are maintained in this website for reference. Placement details, TCEMail, Alumni association and several other links are incorporated in this website. A sample of the website is shown in Fig. 4.

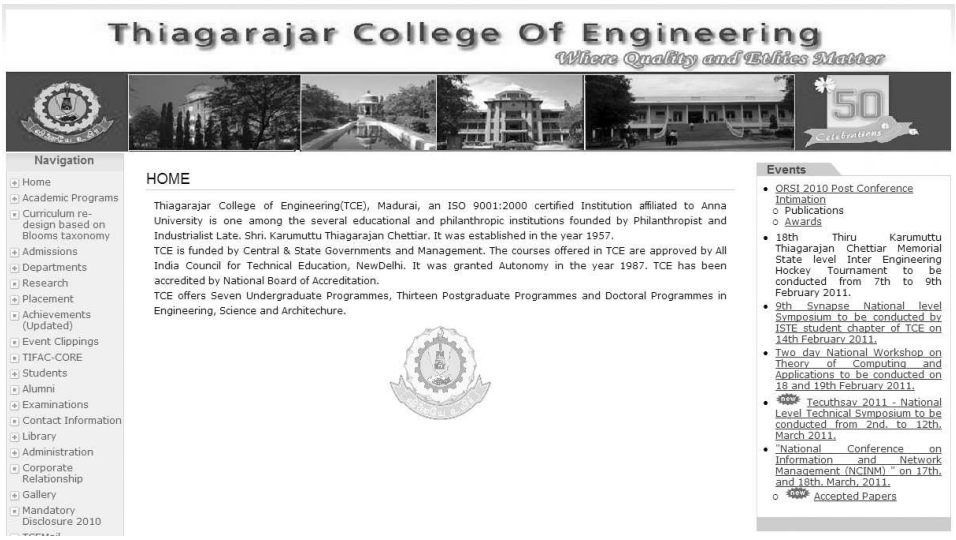


**Fig. 4** *Sample website.*

We considered Departments, Placement, TCEMail, TCENet, and the Library modules for our implementation from the TCENet application. We collected user sessions of the web application, mainly the placement module, for 15 minutes.

The access details for the placement module were instrumented and the equivalent coverage matrix is obtained as shown in Tab. VI.

| P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 |
|---|---|---|---|---|---|---|---|---|---|
| US11 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| US20 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| US31 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| US41 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| US50 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| US61 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| US70 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| US81 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| US91 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| US100 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |

**Tab. VI** *Initial coverage matrix.*

| | P5 | P7 | P8 | P9 |
|---|---|---|---|---|
| US1 | 0 | 1 | 1 | 1 |
| US4 | 1 | 0 | 0 | 0 |
| US5 | 1 | 1 | 0 | 0 |
| US6 | 0 | 0 | 1 | 0 |
| US7 | 1 | 1 | 1 | 0 |
| US8 | 1 | 0 | 1 | 0 |
| US9 | 0 | 1 | 0 | 1 |
| US10 | 1 | 0 | 1 | 1 |

**Tab. VII** *Coverage matrix after essentiality criteria.*

With this coverage matrix as the input, the implementation of the reduction technique is summarized as follows:

| | P5 | P7 | P8 | P9 |
|---|---|---|---|---|
| US1 | 0 | 1 | 1 | 1 |
| US7 | 1 | 1 | 1 | 0 |
| US10 | 1 | 0 | 1 | 1 |

**Tab. VIII** *Coverage matrix after row dominance criteria.*

| | P5 | P7 | P9 |
|---|---|---|---|
| US1 | 0 | 1 | 1 |
| US7 | 1 | 1 | 0 |
| US10 | 1 | 0 | 1 |

**Tab. IX** *Coverage matrix after column dominance criteria.*

The resultant coverage matrix after implementing the essentiality criteria is shown in Tab. VII. The coverage matrix after implementing row dominance criteria is shown in Tab. VIII.

| | **P1** | **P2** | **P3** | **P4** | **P5** | **P6** | **P7** | **P9** | **P10** |
|---|---|---|---|---|---|---|---|---|---|
| **US1** | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| **US2** | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| **US3** | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **US7** | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| **US10** | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |

**Tab. X** *Coverage matrix after adding essential user sessions and pages.*

The reduced coverage matrix after implementing column dominance criteria is shown in Tab. IX. Finally the equivalent reduced coverage matrix is obtained by adding essential user sessions and pages identified using essentiality criteria as shown in Tab. X.

| **Metrics** | **ERC** | | **HGS** | | **Random Reduction** | |
|---|---|---|---|---|---|---|
| | **Original Test Suite** | **Reduced Test Suite** | **Original Test Suite** | **Reduced Test Suite** | **Original Test Suite** | **Reduced Test Suite** |
| **Test Suite size** | 185 | 11 | 185 | 15 | 185 | 100 |
| **Reduction (%)** | 94.05 | | 91.89 | | 45.95 | |

**Tab. XI** *Comparison of ERC, HGS, Random Reduction for TCENet application.*

Similarly, user sessions are collected and instrumented for the other four modules, and reduced user sessions are obtained by implementing the reduction technique (ERC algorithm).This study was carried out in order to evaluate the effectiveness of the user session reduction technique. The data we obtained in the case study actually proved the effectiveness of the technique, since it produces a reduced set of user sessions covering all the EBCPs.

Moreover, we compared the results obtained by our approach with the results obtainable by the HGS reduction technique and the random reduction technique. The results obtained are shown in Tab. XI. From the table it is clear that the proposed ERC technique reduces to 94.05% from the total of 185 test cases compared to 91.89% in the HGS and 45.95% in the random technique. Therefore, we found that the ERC reduction technique produces a more reduced set than the other two reduction techniques.

## 5. Experimental Evaluation

In our experiments, the effectiveness of the reduction strategies was studied by evaluating their fault detection rate. A program under test can be assessed by

counting and classifying the discovered faults. We have used a metric called *fault detection density*, which is a measure of the average number of faults detected by each test case.

**Evaluation Metrics:** For evaluating the reduction techniques, we injected hand-seeded faults.

For example we injected 13 faults and found out which test case identifies which faults as shown in Tab. XII.

We evaluated the test cases with respect to their Fault Detection Density (FDD) and Fault Detection Efficiency (FDE).

| | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 |
|---|---|---|---|---|---|---|---|---|---|
| F1 | √ | | √ | | | | √ | | |
| F2 | | | | | | √ | | | √ |
| F3 | √ | | √ | | | | | √ | √ |
| F4 | | | | | | √ | √ | | |
| F5 | √ | | | | | √ | √ | √ | √ |
| F6 | √ | √ | √ | | | √ | √ | √ | √ |
| F7 | √ | | | √ | √ | √ | √ | | √ |
| F8 | √ | √ | √ | | | √ | √ | | √ |
| F9 | √ | | | √ | √ | √ | √ | | √ |
| F10 | √ | | | | | √ | √ | | √ |
| F11 | | | √ | | | | √ | | √ |
| F12 | √ | | √ | | | | √ | | |
| F13 | √ | | √ | | | | | | √ |

**Tab. XII** *Fault matrix.*

**Fault Detection Density (FDD)** is calculated as follows:

Given a set of test cases $ti$ and a set of faults $f_i$ detected by test cases in $T$, let $fi$ be the number of faults detected by $ti$, then the fault detection density is

$$\text{FDD} = \frac{tf1 + tf2 + tf3 + \ldots + tfn}{|T| * |F|} \tag{1}$$

where $tf_i(I = 1 \ldots n)$ is the position of test case detecting $\text{f}_i$. FDD is the ratio of the sum of the total number of faults detected by each test case and the total number of test cases, normalized to the total number of faults detected. The fault detection density of 1 for a test suite indicates that each test case in the suite detects every fault.

**Fault Detection Efficiency (FDE)** is percentage of faults that has been identified by a particular test suite. The original test suite has the maximum fault detection efficiency. When the test suite is reduced, the fault detection efficiency deteriorates.

$$\text{FDE} = \frac{\text{FDD of the reduced test suite}}{\text{FDD of the original test suite}} * 100. \tag{2}$$

From Tab. XII it is clear that test case t1 detects 10 faults; t4 detects 2 faults and so on. For the original test suite, before applying the reduction algorithm, the FDD is calculated as follows:
Sum of killed mutants of all test cases = 54; Total test cases = 9; Total faults generated = 13

$$\text{FDD} = 54/(9 * 13) = 0.4615385$$

**REDUCTION BY the HGS technique**
Test cases = t1, t2, t4
Sum of killed mutants of all test cases = 14; Total test cases = 3; Total faults generated = 13;

$$\text{FDD} = 14/(3 * 13) = 0.3589744$$

**REDUCTION BY the ERC technique**
Test cases = t1, t2, t3, t4, t5, t6
Sum of killed mutants of all test cases = 31; Total test cases = 6; Total faults generated = 13

$$\text{FDD} = 31/(6 * 13) = 0.3974359$$

When comparing the reduction by the HGS and reduction by the test case minimization using selective redundancy, the results revealed the fact that the fault detection efficiency of test suite minimization using selective redundancy was significantly higher (86.1%) than the fault detection efficiency of test suite minimization done with HGS (77.7%).

# 6. Implementation

The proposed ERC reduction technique was applied to user sessions of several other web applications, namely Portal application, Online Library, Social Networking, Online Shopping.

The users of Portal application will be able to refer to tutorials relating to web development subjects like HTML, XML, CSS and JavaScript. W3Schools is *free of charge* and is funded through text and *display advertising*. The Online Library, called eBookie, provides access to reliable, authorized, published content that went through the rigorous editorial processes of traditional book publishing and then carefully transcribed into the eBook format.

Social Networking includes several social networking websites like Facebook, Orkut, Tagged etc. Users can join networks organized by city, workplace, school and region to connect and interact with other people. People can also add friends and send them messages and update their personal profiles to notify friends about themselves. The Online Shopping called eBay is an online shopping web application that is custom-built for Indian users. Online shopping is the process consumers go through to purchase products or services over the Internet. The access logs for the above web applications were collected and instrumented, and the initial coverage

| S.No. | Web Application | ERC | | | HGS | | | Random | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Original Test Suite | Reduced Test Suite | Reduction % | Original Test Suite | Reduced Test Suite | Reduction % | Original Test Suite | Reduced Test Suite | Reduction % |
| 1. | Portal Application | 135 | 9 | **93.33** | 135 | 11 | **91.8** | 135 | 90 | **33.33** |
| 2. | Online Library | 150 | 11 | **92.6** | 150 | 13 | **91.3** | 150 | 90 | **40.00** |
| 3. | Social Networking | 175 | 11 | **93.7** | 175 | 12 | **93.1** | 175 | 116 | **33.71** |
| 4. | Online Shopping | 150 | 12 | **92.0** | 150 | 14 | **90.66** | 150 | 94 | **37.33** |

**Tab. XIII** *Result of the reduced test suites for subject applications.*

| Name of Application | Number of Test Cases | Number of Test Cases Reduced by HGS | Number of Test Cases Reduced by Random | Number of Test Cases Reduced by ERC | FDE HGS (In %) | FDE RANDOM (In %) | *FDE ERC (In %)* |
|---|---|---|---|---|---|---|---|
| Portal Application | 135 | 11 | 90 | 9 | 72 | 63 | 82 |
| Online Library | 150 | 13 | 90 | 11 | 74.7 | 79.1 | 91.2 |
| Social Networking | 175 | 12 | 116 | 11 | 78 | 65 | 93 |
| Online Shopping | 150 | 14 | 94 | 12 | 69 | 66 | 75 |

**Tab. XIV** *FDE calculation for subject application.*

matrix was obtained. The ERC reduction technique was applied to the obtained matrix in order to get the reduced user session. The results obtained are shown in Tab. XIII.

From the table it is clear that the proposed ERC technique reduces more test cases compared to the HGS and random techniques. For the Portal application, 135 test cases collected were reduced to 9 test cases by applying the ERC technique, whereas they were only reduced to 11 and 90 test cases by the other techniques. When viewing the percentage of reduction from Tab. XIII, for the application Online Shopping, the reduction percentage is 37.33 for the random and 90.66 for the HGS, whereas it is 92.0% using the ERC technique. Thus from the above table, it clear that the ERC technique is better in removing redundant test cases compared to the other techniques, thereby giving a higher reduction percentage for all the subject applications considered.

An analysis based on Fault Detection Density (FDD) and Fault Detection Efficiency (FDE) has also been performed. For the analysis, we injected 13 faults, and based on the formulae given in Section 5, we obtained the results shown in Tab. XIV.

From the table it is clear that FDE of the ERC technique is better than FDE of the other reduction techniques considered. For example, for the online application the fault detection efficiency of ERC technique is 91.2%, but it is only 74.7% for the HGS and 79.1% for the random technique respectively.

Thus the ERC technique is better in both Reduction percentage and Fault detection Efficiency compared to the other reduction techniques. Fig. 5 gives a graphical representation of the reduction percentage and fault effectiveness of the various reduction techniques.

# 7. Conclusions and Future Work

In this paper, a reduction technique, namely the ERC technique, used for simplifying the set of user session data to be considered for testing a Web application, has been presented. This technique is applicable in the context of script-based dynamic web applications and it is based on the assumption that equivalent user behaviors recorded in user sessions can be deduced by analyzing equivalent built client pages that are generated by the WA at run time. Given a preliminary set of user sessions, this approach produces an equivalent subset of user sessions that is able to exercise the same set of user behaviors exercised by the original set. Therefore, it can be used in the first step of a process that designs test suites efficiently for a WA on the basis of user session data. This technique also adds to the quality of the test cases, thereby increasing the testing time efficiency.

This technique has been preliminarily validated by reducing the user session of the TCENet Web application. Our reduction technique is compared with the HGS and random reduction technique to validate its effectiveness. We have proved that our technique reduces more user sessions comparatively, thus increasing the quality of testing and reducing the time, cost and other resources required during testing.

In future work, wider experimentation involving several larger Web applications will be carried out in order to validate the scalability of this approach.

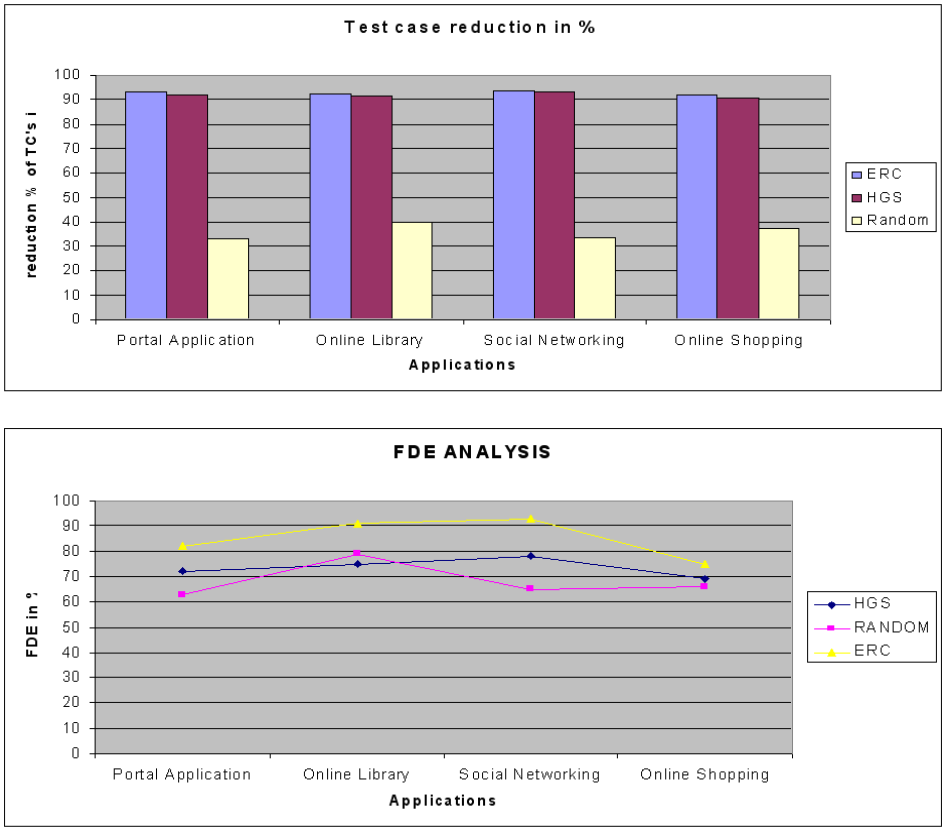


**Fig. 5** *Reduction analysis for subject applications.*

Moreover, we will try to adapt the reduction technique to WAs that are not script-based but implemented with different technologies, and we will combine this method with several testing strategies for exploring the testing effectiveness of using a reduced set of user sessions.

# References

[1] Di Lucca G. A., Fasolino A. R., Tramontana P.: A Technique for Reducing User Session Data Sets in Web Application Testing, Proceedings of the Eighth IEEE International Symposium on Web Site Evolution (WSE'06), 2006.

[2] Antoniol G., Di Penta M., Zazzara M.: Understanding Web Application through dynamic analysis, Proceedings of the 12$^{th}$ IEEE International workshop on program Comprehension, IWPC 2004, IEEE CS Press, 2004, pp. 120-129.

[3] Di Lucca G. A., Di Penta M., Fasolino A. R.: An Approach to Identify Duplicated Web Pages, Proceedings of 26$^{th}$ IEEE Annual International Computer Software and Applications Conference (COMPSAC 2002) IEEE CS Press, 2002, pp. 481-486.

[4] Sampath S., Mihaylov V., Souter A., Pollock L.: Composing a framework to automate testing of operational Web-based software. Proc. of 20$^{th}$ International Conference on Software Maintenance ICSM 2004, IEEE Computer Society Press, 2004, pp. 104-113.

[5] Elbaum S., Karre S., Rothermel G.: Improving Web Application Testing with User Session Data. Proc. Of International Conference on Software Engineering, ICSE 2003, IEEE Computer Society Press, 2003, pp. 49-59.

[6] Elbaum S., Rothermel G., Karre S., Fisher M.: Leveraging User-Session Data to support Web Application Testing. IEEE Transactions on Software Engineering, **31**, 3, 2005, pp. 187-202.

[7] Di Lucca G. A., Di Penta M., Fasolino A. R., Tramontana P.: Supporting Web Application evolution by dynamic analysis. Eighth International Workshop on Principles of Software Evolution, IWPSE 2005, IEEE CS Press, 2005, pp. 175-184.

[8] Harrold M. J., Gupta R., Soffa M. L.: A methodology for controlling the size of a test suite. ACM Transactions on Software Engineering and Methodology, **2**, 3, 1993, pp. 270-285.