ISSN No. 0976-5697

Volume 4, No. 2, Jan-Feb 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

An Intelligent Approach to Automatically test web Applications

Hamideh Hajiabadi*
School of Computer Engineering
Birjand University of Technology, Birjand, Iran
h.hajiabadi@yahoo.com

Mohsen Khani
School of Computer Engineering
Birjand University of Technology, Birjand, Iran
khani@um.ac.ir

Abstract This paper proposes an implemented technique based on model based testing methodology in order to test web application. It is examined on several real web applications. The results confirm the enormous reduction in time and cost consumed by testing process. This work is conducted in three phases. Initially the web application is crawled intended to extract SUT's model describing all of its components. Then the model is trained in order to extract test cases covering all of the SUT's paths. Eventually test cases are executed and the results are reported. This work is a sort of black box testing technique intended to automate testing process as much as possible. Several ontologies are employed in order to eliminate human's role.

Keywords: Ontology, web mining, model based testing, dynamic analysis

I. INTRODUCTION

The number of web based software developed and locates on the internet for public uses increases greatly; therefore, testing becomes critically important to developer in order to compare real system's behavior with supposed one. There are several kinds of testing used in industry like quality assurance, stress test, load test, functionally test and etc. the testing process involves in generating variety of test cases in order to cover expected criteria. Then the generated test cases are examined, and a report containing the results of each test case is produced. [1]

The testing process aims to discover faults in the system under test (SUT). It is necessary that sufficient test cases are produced which cause tester convinced that the system has approximately no fault. This sort of testing conducted manually sounds exhaustive. A test case includes test data and expected results. Criteria coverage is used to generate test cases systematically. The test cases covering those criteria well, the test case generation process is stopped. In a web based application test case is considered as a sequence of URL and several data provided for URL's parameters. [2]

Focusing on the tremendous amount of costs consumed by manually testing, the automated testing is regarded as an importance, and it seems necessary to be replaced traditional testing with more automated ones. Model based testing aims to automatically drive test cases from the SUT's model. There are several challenges involving in model based testing, some of which are mentioned as follows. [3]

a. Models which are completely matched with SUT should be provided.
b. A certain amount of real data must be supplied after deployment.
c. Weather the test cases succeed or fail should be automatically comprehended, and it is concluded by comparing expected results with real ones.

Consequently it can be concluded that the troubles latest model-based testing methods are involved in raise; because, the rate of dynamics latest systems included increase enormously in comparison to the past.

The rate of changes done on the web-based applications is extremely great ;therefore, in many cases developer directly makes changes in source code and does not follow software engineering process; hence, it can be considered that there is no model completely adapting with code. Consequently it is seemed crucial the first job is extracting SUT's model from source code. [4]

This paper proposed a model based testing technique to test web application based on the model extracted by reverse engineering technique.

First the category of web application is recognized by using data mining techniques. it being mentioned earlier, test cases for web based application contains a sequence of URL and some data provided for URL's parameters. This URL sequence is generated by utilizing some model traversing methods, and to produce suitable data automatically, the category of web based application is utilized to select proper ontology, and the ontology facilitate producing data automatically.

This paper is organized in 6 sections. In section 2 varieties of related works are reviewed and some of their drawbacks and profits are briefed. Section 3 focuses on how automatically categorize a web page and how practical it is to improve automated testing. Modeling a web application is demonstrated in the next section. Details of testing processes are discussed in section 5. The proposed technique is examined on several real web application and the results are presented in section 6. Section 7 includes conclusion and future works.

II. RELATED WORK

In these years this topic has fascinated increasing number of researchers as it can be proved by the great number of workshop specialized to this subject. Among these escalating number of papers performed on test and analysis, the only ones concentrate specifically on model based testing of web applications and employ black box techniques are selected to be briefly explained.

Liu and his friends [5] developed a tool named WebTestModel which is a kind of data flow diagram. He produced test cases from examining the model, each object of which is a web application's component.

Ricca and Tonella [6] proposed a model consists of two tools: one for modeling a web application and the other for

testing purpose. The modeling tool which is a kind of reverse engineering technique mine SUT in order to extract several UML models. The models already extracted utilized by testing tools based on MBT techniques. The only drawback of their work was the amount of manually attempt demanded.

Andrew and his friends [7] developed a tool named FSMWeb mining SUT so that produce several Finite State Machine Diagrams (FSMs) which are leveled. In the lowest level each state is a page of the web application. Each diagram is considered as a node of the higher level diagram. In his model a test case is regarded as a sequence of state and the parameters required.

Bertolino [8] conducted a survey on the testing processes and addressed the challenges involved in. she demonstrated a variety of researches conducted before and discussed the interesting continuing works. She represented the starting point to researchers wanting to initiate working in this field.

Wang [9] proposed a method to generate test cases by analyzing source code. The information he extracted from the canalization consist of input parameters, domain information and the user navigations. In spite of being the perfect model in producing test cases, it is limited to a certain programming language and techniques.

III. CATEGORIZING THE WEB APPLICATION

The most important challenges arisen through the web application testing process are selecting data for the demanded parameters. Informally it can be stated that filling forms is the task which is so complicated to be automated.

Considering the same fields must being filled in two different kind of web application, the valid data for those might be thoroughly different. Consider a field demanded the same thing likes *age* in two different kind of web application for example: an educational system and a shopping system. In the first one the valid data is between 18 and 45, but in the other one the valid range is from 5 to 110. In order to produce data correctly, the category of the web application should be already specified.

There is a work extracting the category of web application automatically using some sort of data mining techniques [10]. This work is utilized to automatically identify the subject of the web application.

It having be earlier explained, several ontologies are employed to create test data automatically. With the web application's category being recognized, the proper ontology is accurately selected in order to generate test data more specifically. The details of these steps are expressed in the following sections.

IV. WEB APPLICATION MODELING

The most important way to automate testing process is the model based testing technique. This kind of technique drives test cases from several models the SUT is based on. The most important thing makes web based applications totally different from other kind of applications is the way they are developed and modified. the rate of changes done on the web based applications is considerably more than the others; moreover, developers do not pursue development process and immediately do the changes on the source code and ignore updating models; therefore, it can be emphasized that there is no model completely illustrating the SUT. So in order to use model based testing for web based application, it seems necessary to extract the models in any way.

In order to extract model from SUT, there are two kinds of techniques can be followed: the black box technique and the white box one. In the second one the source code is mined in order for extracting the model of the system. Although this model is completely precise and contains all of the SUT's paths, it is restricted to the techniques SUT is developed by, and it is not practical on every web based application. The black box technique considers the SUT as a black box, and the details of implementation are hidden from the tester's eyes. The proposed work employs black box technique in spite of white box one.

The web applications include several pages and the links between them. Although sometimes the content of these linked pages are static and invariable through navigating, some others like forms make dynamic pages and their contents are differed by information gotten from users. A form being submitted, the gathered information is sent to the server side application, and a dynamic page is shown to the user as a reply.

There are two kinds of links between pages: the navigations made by active component and the hypertextual links. The first kind of links divides into several parts. First part includes the navigation produced by forms submitted to the server side application, and a dynamic page is sent back. Second part consists of the navigations based on redirection between a client script and a client module. In order to model all of the paths, the parameters used in the forms must be produced intelligently considered as a most complicated challenge in automating web application test.

A. Static analysis:

This phase is a sort of reverse engineering process intended to model all of the pages, its components and the links between the pages. All of the fields and their types are carefully extracted and kept in the data base. The data base utilized for this purpose is a kind of Mysql. In order for modeling the entire components, Htmlunit[1] a java open source library is utilized. It simulates a browser and executes objects. After a page is analyzed and its components are entirely extracted, one of the objects is orderly selected. If the chosen object includes:

a. A hypertextual link: the onClick event handler is invoked, and the simulated browser navigates to the new URL.
b. A form: it is filled and submitted. Then the browser peruses new address.

It being mentioned earlier, in order to reach all pages hidden behind the forms, variety of data should systematically be selected. In traditional evaluations a tester manually attempt to provide data so that all of the pages behind the forms are covered. This sort of testing is time and cost wasting.

B. Filling forms:

It can be stated briefly that several ontologies are employed to fill forms automatically and the hidden parts of web application are achieved and get modeled. Jena[2] an open source java library is utilized in order to communicate with several ontologies.

In order for collecting a set consisted of parameters demanded to fill by users in the forms, a tremendous number of forms are downloaded from live web applications. The parameters divided in 6 parts: personal information, address information, bank account information, login information and email information. An ontology called general ontology consisting of those classified information is manually build.

To attach restriction to each concept in the general ontology, the category of web application should be first identified.

The proposed tool concentrates on 5 types of web application: 1- shopping sites 2- news 4- scientific 5-others. In the earlier sections, the way web application is categorized is fully explained. Web application's category being identified, some restrictions are manually attached to the concepts in general ontology.

A form's component being demanded to fill, a mapping to a concept of general ontology is performed. That concept and its restriction are utilized to generate data automatically. For example consider the *age* concept with two attached restrictions: *min* and *max*. In order to generate valid data automatically, a digit between min and max is randomly selected.

Some of form's parameters cannot be produced randomly like *usernames* and *passwords*. These sorts of data are kept permanently in the data base developed with the SUT. The valid test data for these parameters are those which are kept in the DB and the invalid test data are those which are not saved; hence, the data stored in the DB must be accessible. This sort of testing technique allowed to access stored data is called gray box testing. The proposed technique is a kind of gray box test.

There are several works conducted to convert a relational DB to ontology. Etminani in 2009 offered a technique converting a relational DB to ontology. Its technique is implemented with java and specifically Jena open source library. The proposed model follows technique she produced to convert DB to an ontology which is here named DBConverted. [11]

In order to generate test data for a form's parameters, in the first step a mapping to DBConverted ontology is done. If there is no such a possible mapping, in the second step a mapping to general ontology is automatically planned. Then some valid data are generated automatically in order for passing forms and model hidden parts of web application.

In this phase all of the SUT's parts are fully modeled and it is prepared to extract test cases from the model.

C. *Test cases generation and execution:*

It being briefed earlier, a test case for web application contains a sequence of URLs and several required parameters should be carefully provided. The extracted model can be considered as a graph. In order to generate a sequence of URLs automatically, the graph is traversed which covering all of the SUT's paths.

Two ontologies fully explained in previous sections are utilized to generate test data automatically. Wordnet, predefined lexicon ontology, is employed to make mapping process a lot easier. By considering a text box in the form expected to fill, it is planned to map the text box onto a concept included in DBconverted ontology at the first attempt. If it is failed, a mapping onto general ontology's concept is conducted.

With the mapping being perfectly done, several test data should be generated automatically. Boundary coverage policy is used to specify how data should be selected.

If the concept's type is digit, Min and max restrictions are employed to generate six test data including min, min-1, mid, max, max+1 and a value out of range. If only one restriction is attached to the concept, three produced data includes: one data inter range, one data in the boundary and one another out of range.

If the concept's type is text and it is added max length restriction. One text that is shorter, one another longer and a text containing special characters are selected. Following paragraphs states the way several types of form's elements are treated.

a. ***Text area:*** it is attended to get some information in the text format from users. The only constraint attached is max length; therefore, as it is explained earlier, three texts are produced.

b. ***Radio group:*** one radio is randomly clicked.

c. ***Check box:*** there are two states: checked and unchecked, one of which is randomly selected.

d. ***Select:*** this variable can get a few numbers of options definitely exhibited, one of which is randomly selected.

D. *Test case execution:*

Being generated, test cases should be executed and the results must be reported. Selenium[3] is a record and replay tool recording a test scenario. A variety of test data can be manually supplied for recorded test scripts. Afterward they are executed by Selenium and the results are exhibited.

In order to automate this phase, test scripts generated by this proposed technique are in the same format Selenium can execute. Generated test scripts are executed by Selenium automatically. It makes executed test scripts distinguished by assigning green and red color to each. Figure 1 illustrates a test script generated by proposed model in order to fill and submit the attached form. This test script can be executed by Selenium.

```
<table cellpadding="1" cellspacing="1" border="1">
<tr><td rowspan="1" colspan="3">2</td></tr>
<tbody>
<tr>
      <td>open</td>
      <td>/educ/exam/index.php?Lan=En</td>
      <td></td>
</tr>
<tr>
      <td>type</td>
      <td>id=NID13</td>
      <td>2733465378</td>
</tr>
<tr>
      <td>type</td>
      <td>id=TrackNumber13</td>
      <td>8237283</td>
</tr>
<tr>
      <td>click</td>
      <td>css=td &gt; input[type=&quot;button&quot;]</td>
      <td></td>
</tr>
</table>
```

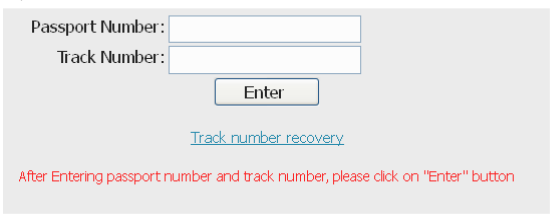

Figure 1 a sample of test script

E. Evaluation and results:

Figure 2 demonstrates the testing processes manually conducted in traditional testing. Proposed tool tries to automate the processes as much as possible.

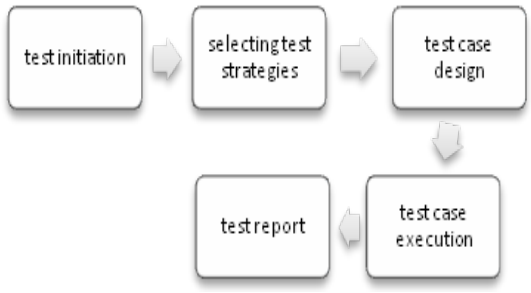

Figure 2 testing processes

Three complicated and time consuming steps are test design, test case execution and test reports all automated in the proposed model.

Table 1 comparing proposed model with several traditional models

Existing tools	Test Design	Test Execution	Test report
Manual testing	manual	Manual	manual
Record/Replay tools	manual	automatic	automatic
Proposed model	automatic	automatic	automatic

Table I compares proposed model with traditional testing and record/replay tools. Record/replay tools are the suite of testing tools recording a test script manually and replays recorded test scripts with several test data provided manually.

The proposed model is examined on a number of web based applications, and some of the results are demonstrated here.

The address of the International Branch of Ferdowsi University of Mashhad's web site is http://pooya.um.ac.ir/educ/exam/index.php?Lan=En which is utilized to be examined by proposed model. The forms included in this web site being the sort of complicated ones, it is selected to be examined by the proposed model.

There are four steps performed during analysis: initially the web application is mined to be categorized, and *academic* is the output of this step. Then general ontology is manually enriched by adding adequate restrictions to its concepts. Data base is transformed to an ontology named DBConverted next. The java implementation tool is examined to the web site in order to produce test scripts in the following step. Eventually generated test scripts are executed by Selenium and the report is exhibited.

It being explained earlier, the static analysis and test is completely done automatically; hence, the only thing which is aimed to be conducted automatically is filling forms. As Table II is shown, although the forms contained in this web site are rather complicated, the average automation degree achieved by assumed model is incredible.

Table 2 The results obtained by examining Ferdowsi University's web site

Subject	**Result**
Average Automation Degree in Filling Forms	84%
Path Coverage	78%

F. Limitation and Future Work:

Making promotion in automated web application testing, proposed model still encounters a number of open source questions. With escalating progress in semantic web applications, it seems necessary that future works concentrates on semantic web applications.

V. REFERENCES

[1] G. Eason, B. N. (1955). On certain integrals of Lipschitz-Hankel type involving products of Bessel functions. Phil. Trans. Roy. Soc. London, vol. A247 , 529-551.

[2] B. Michael, F. J. (2002 May). VeriWeb: Automatically Testing Dynamic Web Sites,. IWWWC.

[3] D. Kung, C. H. (2000 July). A model-based approach for testing Web applications. In Proc. Of Twelfth International Conference on Software Engineering and Knowledge Engineering. Chicago.

[4] Carlini, U. D. (2002). WARE: a tool for the reverse engineering of Web applications. Proceedings of the Sixth European Conference on Software Maintenance and Reengineering CSMR-02.

[5] C. Liu, D. K.--. (2000). Structural testing of web applications. 11th IEEE International Symposium on Software Reliability Engineering (pp. 84--96). IEEE.

[6] Tonella, F. Ricca. (2001 May). Analysis and testing of Web applications. In 23rd International Conference on Software Engineering (ICSE `01), (pp. 25-34). Toronto.

[7] A. Andrews, J. O. (2005 July). Testing Web Applications by Modeling with FSMs. Software and Systems Modeling, (pp. 326-345). German.

[8] Bertolino, A. (2007). Software Testing Research: Achievements, challenges, Dreams. IEEE conference, Future of Software Engineering (FOSE'07).

[9] M. Wang, J. Y. (2008). A Static Analysis Approach for Automatic Generating Test Cases for Web Applications . International Conference on Computer Science and Software Engineering, (CSSE.2008). Wuhan, China.

[10] H. Hajiabadi, M. H. (2011). Extracting Specific Category of Text Documents Using Ontology. 2011 IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE) (pp. 489 - 491). Penanag, Malasia: IEEE.

[11] K. Etminani, M. K. (2009 July). Transforming Relational Databases to the Corresponding Ontologies. Networked Digital Technologies (NDT 2009),. Ostrava: the Czech Republic.