An Automated Model Based Approach to Test Web Application Using Ontology

Hamideh Hajiabadi, Mohsen Kahani
hajiabadi.hamideh@stu-mail.um.ac.ir, kahani@um.ac.ir
Computer Engineering Department, Ferdowsi University of Mashhad

***Abstract*— Nowadays the growth of web application development is great; every day a variety of new web applications are raised on the Internet for public use. Web applications have n-tier architecture, so the server side programs could change without client interference and this process could be done more times. Consequently the testing is the important issue in the web application development. This paper proposed an automated model based testing technique to test web application from its structural model. Firstly using reengineering approaches the structural model is constructed to demonstrate static aspects of the web application. Then using several ontologies and mapping tools, test cases for filling forms are automatically generated to model and evaluate dynamic features of the web application. The technique implemented as MBTester tool and applied to a few web applications. The results presented in this paper indicate the dynamic attained by MBTester is great.**

Keywords—boundary coverage criterion, evaluate partitioning, test case, ontology, mapping

I. INTRODUCTION

Web application developing has become a significant area, which has a great progress among other fields of software engineering. Web application has essential characteristics which have made it different from other ones. Web applications consist of some modules and interactive components in which various technologies could be used independently. It is important to note that in addition to the growth of web applications, the techniques used in web application developing have also great growth. So this fact makes web application testing more difficult.

Some kinds of complexity are involved in web application testing. One of these is its heterogeneous nature. Web application consists of several components implemented by different technologies and programming languages. Dynamism existence is the other complexity essence in testing of web applications. Web applications consist of static and dynamic components. Static components can be automatically tested; Spider like tools could do it automatically. Users submit their inputs by using form components through crawling and the output page sent back in response to a submitted form is always dynamic. The content of dynamic pages varies according to user inputs. No tool exists for testing the dynamic aspects of web applications automatically. The only sets of tools which are able to test the dynamic aspects of web applications are some sorts of record and replay tools. They initially record a testing scenario, and then generate test script for testing assumed scenario, the recorded test scripts are used for automatically regression testing. But a negligible change in GUIs could make many test scripts useless. The other drawback of this set of tools is their huge cost due to the manual recording of test scripts. As a result, automation and proper coverage are the key issues in the testing of web applications.

In this paper, an automated technique is proposed that extracts a structural model of web applications based on client side requests and responses that are automatically generated by the programs run on the server. The difficult part of testing processes is producing values to fill forms. Any sort of values covers some contents of web applications. In that case to cover all aspects of web application, a lot of values should be generated sagaciously. These values are obtained using ontology. The boundary coverage and equivalent partitioning criteria are used in generating test data. The web application deals with the information that user enters in a different way. Some parts of information which are more important than the other parts are kept in the system database but the other parts which are less important need not be kept there in general. The degree of dynamism and coverage in this scenario is higher in comparison with related works.

Section 2 presents and discusses the works which exist in the model based web application testing. The techniques used in the reverse engineering process are explained in section 3. Section 4 illustrates the analysis model in detail. The testing skills used in the proposed model are specified in section 5 .In the conclusion section, a summary of the paper and future works are presented.

II. RELATED WORK

A model based test generation technique was proposed by Kung, Liu, and Hsia [1, 2]. The models used in their technique are Object Relation Diagrams, Object State Diagrams, a Script Cluster Diagram, and a Page Navigation Diagram. The technique proposed by them uses source code in the model generating and testing process. The techniques which use source code in their processes are not always such general ones to be applied to the whole sort of the web applications, whereas our technique does not make any use of source code.

Lee and Offutt [3] proposed an approach in which test cases are generated based on mutation analysis. Their approach is based on XML based interactions and only tests the interactions among modules, whereas our approach tests the whole system. The automation degree provided by their technique is not considerable.

Liu et al. [4] observed each component of web application as an object. He exploited the data flow between these objects and generated test cases based on them. He called his tool WebTestModel.

Ricca and Tonella [5] have developed ReWeb and TestWeb tools. They used source code analysis of processes and

exploited a sort of UML class model to symbolize components of web applications and their transactions. Modeling phase is done by ReWeb semi automatically in two steps: modeling static features and dynamic features. The second step of the modeling always deals with the forms and the pages hidden behind the forms. The data used to fill forms is selected manually by tester. We could informally say that the difficult part of the modeling is done manually and they only automate simple parts. TestWeb uses the model extracted in the modeling phase then using source code coverage criterion, test cases are generated.

Andrew, Offutt and Alexander [6] proposed FSMWeb that represents web applications using hierarchal Finite state machines. At the bottommost level each node of the FSM is a page. At the middle levels the nodes are the FSMs of the ones in their bottom level. A test case is a sequence of states with some parameters needed for each state. The test cases of the bottomed level are joined to create top level test cases.

Wang, Yuan, Miao and Tan [7] developed a test generation method based on source code analysis. It means that source code of application is analyzed to extract interfaces which are composed of: input parameters, domain information and user navigation map. Because of source code analyzing, it was limited to specified programming language and does not work with new technologies.

III. REVERSE ENGINEERING WEB APPLICATIONS

Web applications are always developed without following the software life cycle processes, the requirements are not recorded, and the models are not designed. The developer early enters the implementation phase. Then we do not expect developer to present a proper model for testing purpose. Subsequently such a model should be produced from artifact. Reverse engineering processes are in fact a way to present the abstract demonstrations of the implemented work. These demonstrations provide useful information in a diverse level of granularity. The pages are the coarse grained part of the structure of web applications. Each page includes different components which are split to activate and inactivate components. The pages are divided into server pages and client pages. The server pages are set up on the server and the pages sent back to a client requests are client pages which are categorized in two groups as static pages and dynamic pages. Static pages are those that their contents are fixed and stored in a persistent way and the dynamic pages are built on the fly and their contents are dependent upon the inputs entered by users. [9] Exploiting the structure of web applications could be done in two ways:

- White box testing technique or source code analyzing: The tools implemented by this way are restricted to some programming languages and web technologies. Because of large growth of web technologies it should be developed and maintained all the time. If it had not been maintained, after a few times its interoperability would be lost and will not work with new coming technologies.
- Black box testing or client side analyzing: this technique is done on the client side; the structure of the web site obtained by making requests and examining responses. It is important to note that dynamic and static pages couldn't be distinguished by this way. But we can informally say when a form is submitted, the target page is dynamic and its content is dependent on data which are entered in the form. We use this technique in our approach.

Our approach is a kind of gray box testing tool, so we could make use of the system database and employ the data kept in it to fill forms. One of the ontologies used in this project is the system ontology, which is learnt from the system database. Another ontology which is employed is built as follows: More than a hundred forms are downloaded from different web sites. Afterward, their fields are categorized and synonym words are eliminated using the WordNet[1] ontology. The result is about 70 words remained, which are utilized for constructing the ontology. We refer to this ontology as general ontology. General ontology is equipped with defining accurate constraints. We use general ontology and its constraints to produce data which are not kept in system database.

The resulting model is even computable in the presence of high extreme dynamism. It is also obtained with a high degree of automation in the dynamic behavior of web applications. If the inputs used in this generation could cover all relevant behaviors of the web application, the model generation is completed. This is an intrinsic limitation of all dynamic analyses. The tool we propose to support the recovery of structural model from client side analyzing is exhaustively described in the following section.

IV. ANALYSIS MODEL

Web applications are composed of some pages and the navigation links between these pages. A page includes information which is exhibited to user and some links to another pages. The model proposed here is a kind of navigation model that demonstrates the navigation and interaction pattern of web application. Links to the other web sites are assumed as external links. External links are ignored in the modeling process. [5]

A web page contains some information and any number of HTML elements like forms. Each form includes some input elements which are filled by users. The data provided by user through navigating are gathered and submitted to the server. The target of the submit link is always a dynamic page. The dynamic pages are different in respect to the input values, and they may differ from one value to another. Suppose a form that gathers user interests, and shows additional information about his interest. For example, if user interest is sport, it shows a page containing additional information about sport. Consequently, for exploiting all paths of web application, it should produce all of the values that input variables could get. Web exploiting process is divided into two phases, as follows:

- Phase 1: Static analysis of web application. In this phase static structure of web application is constructed containing static pages, static links, etc.
- Phase 2: Dynamic analysis, in the way just explained. Since dynamic pages are hidden over the forms, in this phase input

[1] WordNet is a large lexical database of English, developed under the direction of George A. Miller. http://wordnet.princeton.edu/

values of the forms are produced and submitted to yield dynamic pages. Informally, in this phase we proceed to Automatically Filling Forms.

A. Static Analysis

It initially starts from a pre-given URL; the states reachable from the starting page are examined sequentially. For each page the set of possible actions are attained and kept in the database. Current page is scanned by another component to analyze it statically, if an error is discovered; it is logged for later process. If a cycle is detected, the exploration from the current page is ignored. The actions which are saved are consecutively investigated and inspected in a manner explained later. After executing each action, if no error encounters, a page is properly returned from the web server. If for some reason, like a breaking page or something, an error occurs, that error needs to be reported and kept in a persistent way. Otherwise the algorithm is recursively reapplied to the returned page.

After analyzing a page and selecting an object, the selected action is executed. In the proposed approach we make use of an open source HtmlUnit[2] library for simulating a browser and executing objects. The manners are employed in respect of objects are explained bellows:

- If the action is a link, the onClick event handler (if any) is invoked and if it returns true, the browser moves to the HREF of the link.
- If the action is a form, the inputs to fill form are selected in a manner to be explained in the next section. If it has an onSubmit event handler, the relevant handler is invoked and if it returns true the browser is relocated to the form's *action* parameter.
- If the action is an active form element, its related event handler is invoked. It is important to note that this process is performed in the next phase, form filling.

[10]

B. Automatically Form Filling

Difficult in the dynamic web application automated testing process is dealing with forms. Some aspects of dynamic web applications are hidden behind forms. Consider a form which gathers user interests and show additional page with auxiliary information about his/her interests. This page is dynamic and its content varies by user interests. In order to explore all the pages which are hidden behind the forms, a variety of input values to cover all the states should be sensibly produced. Consider an educational system that students enter in with *userId* and *password.* Without auxiliary information no tool can explore successor states of such a form except small states which are built by submitting incorrect *userId* and *password.* Therefore additional information that kept in persistent location should be used by the tool in order to crawl through forms.

A variety of data that a user enters in the forms is important and should be kept in persistent places such as databases. Some other parts of the data are not as important, and hence are temporarily saved and used for processing. This persistent data consist of the mentioned auxiliary information required for crawling web applications. This auxiliary information could be selected by human tester who can access to database, of course it is done manually, but the question is how to fill the forms automatically with the persistent data? Moreover, how to map one input variable on a form to one filed on the system database? The next section addresses these problems.

C. Using Ontology

In this project, we exploit three ontologies, which are introduced as follows:

- General ontology: Most of web sites collect similar information from users like personal information, bank accounting information, educational information, and so on. We believe that nearly 80 percent of the collected information is categorized as general. So we bother to collect this information from a variety of web sites. In that case, we download nearly one hundred forms from various web applications which are available in the internet. Then, textbox elements included in the form are detached and the names of their corresponding variables are stored in the specific file. Then, we try to eliminate synonymous words. After eliminating synonyms, the file contains words that none of them have similar meanings. Afterward, general ontology is constructed from the remaining words. Its concepts are restricted with some accurate constraints.
- Lexicon dictionary like WordNet: This ontology is utilized to help in understanding synonyms and antonyms. WordNet is a large lexical database of the English words.
- System ontology: This ontology is learnt from the system database. We use the rules which are explained in [9] for transforming database into ontology. The data stored in the database are converted to the individuals in this ontology.

We make use of the three mentioned tools to fill forms automatically. Consider one text box element which should be filled: To assign a value to a corresponding variable of the text box, the tool firstly tries to map the variable to one of the concepts available in the system ontology. If it is done, several test cases are examined from its individuals included in system ontology. Otherwise, it tries to map the element to one of the concepts presented on the general ontology. Afterward, the constraints existing for mapped concept in general ontology are used and some values are generated. It is important to note that generation is done based on boundary coverage policy.

For better explanation, suppose the age concept on the general ontology. It has two constraints of min and max, which respectively represent minimal and maximal value that age could get. Then, Boundary coverage criterion is used to produce some values for assigning to the age variable.

In the proposed tool boundary coverage and equivalent partitioning criterion is used for generating test cases. Considering an interval [min,max]. After applying the above criterion, test cases will be min, min+1, mid, max, max+1. In short, for automatic form filling, the database is exploited for the data existing on it. On the other hand, the general otology is used for the missed data, and hence using its constraints and some data coverage criteria, a value is generated.

[2] HtmlUnit is a "GUI-Less browser for Java programs". http://htmlunit.sourceforge.net/

TESTING

Verification and validation are two common objectives of process testing. Testing process is divided into dynamic mechanism and static one. The techniques used in the static mechanism are different from those of dynamic one. Given an HTML page, the static testing inspects the source of the page and discovers possible faults. In the dynamic testing a vector of input values are examined and executed on the system under test (SUT) and the obtained results are compared with expected results. If diversity is detected, it is reported.

A test case for a Web application includes sequence of URL and the input values needed for each page which contains forms. The sequence of URL is the path selected from the structural model. The sequences of URL are selected by applying some graph traversing algorithms to the structural model. Input values needed for each URL are chosen by applying the boundary coverage and equivalent partitioning policy. Thus the path selection is independent of input values and it could be automatically done. For that reason, the difficulty existence in the testing processes is made by dealing with the pages contains forms. Execution consists of requesting the Web server for the URLs in the sequence and storing the output pages.

The proposed approach uses the structural model to define URL sequence by using some model traversal algorithm based on some model coverage analysis. Then, three ontologies are used to create input values for defined URLs. Boundary coverage and equivalent partitioning policy is used for test case generations. The format of test scripts which are generated by the tool is like the script's format that could be executed by Selenium [3]. Selenium is an open source record/replay tool which is used in the tool for executing tests.

It is important to note that test script also contains oracles and Selenium could give us the number of test steps that are failed or succeeded.

Conclusion and Future Works

We have presented and quantified a new approach for testing web applications. This new approach differs from existing approaches in generating test cases.

Firstly a structural model of a web application is obtained. Then test cases which verify the restrictions are created using several ontologies. At the final steps Selenium is used for executing test scripts and assigning verdicts. The analysis and testing techniques proposed in this paper were successfully applied to several real world Web applications.

The MBTester technique still has a number of open questions and issues. Current work is largely focusing on automation and evaluation. The tool is in a preliminary stage and works are undergoing to remedy its shortcomings.

REFERENCES

[1] D. Kung, C. H. Liu, and P. Hsia. "A model-based approach for testing Web applications". In Proc. Of Twelfth International Conference on Software Engineering and Knowledge Engineering, Chicago, IL, July 2000.

[2] D. Kung, C. H. Liu, and P. Hsia. "An object-oriented Web test model for testing Web applications". In Proc. of IEEE 24th Annual International Computer Software and Applications Conference (COMP-SAC2000), pages 537{542, Taipei, Taiwan, October 2000.

[3] S. C. Lee, J. Offutt. "Generating test cases for XML-based Web component interactions using mutation analysis". In Proceedings of the 12th International Symposium on Software Reliability Engineering, pages 200{209, Hong Kong China, November 2001. IEEE Computer Society Press.

[4] C. Liu, D. Kung, P. Hsia, and C. Hsu. "Structural testing of web applications". In Proceedings of the 11th IEEE International Symposium on Software Reliability Engineering,pages 84--96, Oct. 2000.

[5] F. Ricca and P. Tonella. "Analysis and testing of Web applications". In 23rd International Conference on Software Engineering (ICSE `01), pages 25{34, Toronto, CA, May 2001.

[6] A. Andrews, J. Offutt, R. T. Alexander. "Testing Web Applications by Modeling with FSMs", Software and Systems Modeling, German, July 2005, pages: 326-345

[7] M. Wang, J. Yuan, H. Miao, G. Tan: "A Static Analysis Approach for Automatic Generating Test Cases for Web Applications", 2008 International Conference on Computer Science and Software Engineering, (CSSE.2008), Wuhan, China

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

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

[10] B. Michael, F. Juliana, G. Patrice. VeriWeb: Automatically Testing Dynamic Web Sites, In: IWWWC. (May 2002)

[3] Selenium is a suite of tools to automate web app testing across many platforms. http://seleniumhq.org/