American Journal of Software Engineering and Applications
2014; 3(5): 63-67
Published online November 20, 2014 (http://www.sciencepublishinggroup.com/j/ajsea)
doi: 10.11648/j.ajsea.20140305.12
ISSN: 2327-2473 (Print); ISSN: 2327-249X (Online)

Model-based approach to design web application testing tool

Dalila Souilem Boumiza[1], Amani Ben Azzouz[1], Salma Boumiza[2]

[1]Applied Computer Science Department, National Engineering School of Sousse, Sousse, Tunisia
[2]Computer Science Department, Technical University of Sofia, Sofia, Bulgaria

Email address:
Dalila.souilem@yahoo.fr (D. S. Boumiza), amanibenazzouz@yahoo.com (A. B. Azzouz), salma.boumiza@yahoo.com (S. Boumiza)

To cite this article:
Dalila Souilem Boumiza, Amani Ben Azzouz, Salma Boumiza. Model-Based Approach to Design web Application Testing Tool. *American Journal of Software Engineering and Applications*. Vol. 3, No. 5, 2014, pp.63-67. doi: 10.11648/j.ajsea.20140305.12

Abstract: Software engineering is a systematic approach defined as a science of industrial engineering that measures the practical methods and working process of the software engineers. This approach is based on analyzing, designing, assessment implementing testing and reengineering processes of given software All those phases are very important and have a specific role in SE's cycle, especially software testing that acts as a significant element in this cycle and it represents also a fundamental key for software quality assurance. Software testing has as goal to test the software performance by measuring the gap between the expected behavior of the software under test and the test results. This comparison allows the tester to analyze errors and bugs in order to fix them and develop the software performance. As a critical factor in SQA, software testing is considered like a definitive review of the tool's specification: it permits the tester to redesign the tool specification after the test in case of failure. This procedure is also applied on web applications, in similar ways to obtain the same goal: applications quality assurance, but the web applications are more complicated to be tested because of the interaction of the application with the rest of the distributed system. In fact, in more precisely terms, web application testing is a process that measures the functional and non functional proprieties of a given web application to analyze its performance in order to fix errors or even to reach a better level of the application under test. The demand on web applications or generally on software testing tools groups up with the increase in applications or software failures and cost.

Keywords: Software Testing Tools, Software Quality Assurance, Web Application, Model-Based Testing

1. Introduction

The main idea of this paper is trying to improve the popular existing approach of testing web applications. In fact the most used technique to test web applications is the approach record and replay but we have proved that it is an unreliable technique so that an alternative solution must be taken. This settlement is the model based testing technique, and our project will be based in this fundamental perspective.

In fact, model based testing tool is an existing solution, but is it still a new approach that just some tools support. Our goal is to develop a tool that supports this technique and at the same time is simple and easy to be used by a regular user thanks to a graphical user interface GUI.

Our GUI will form a simple way to users to test web scenarios through the new "model-based testing" technique introducing a new and a simple way to perform test cases.

2. Record and Replay Approach [1]

This technique, as its name indicates, consists in recording a scenario from a web application then trying to replay it again in order to extract failure cases and to ensure right ones.

A web scenario is a group of a finite number of actions that the user can do manually. This group includes any action that can be performed by the mouse from a web page displayed in a browser like: clicking a button, clicking a link, filling a text field, dragging and dropping

The Scenario is recorded step by step, which means action after another, to form a test case. After being registered by the tool, this test case can be replayed so that the user can observe the generated results after the test execution and analyze them.

Most of web testing tools apply the "record/replay" function to create a test scenario, by following those steps: The user opens the Website to be tested, activates the record

function, navigates into the website and select the desired actions, stops the record

The program saves the recorded actions as a file that can be opened at any time to replay the scenario.

3. Model-Based Testing [2] [3]

The model-based testing approach is not just specific for web applications only, but it is generally designed to fit all sorts of software.

Model-based testing is the testing process proceeding from a model that describes the comportment of the application that the user wants to test.

This model will be used in order to represent and simulate the behavior of the system under test (software, web application…).The model must be abstract to be faithful to the real system description. The key skill that is really required to make the model successful and derive from it good test cases is the design of a good and abstract model that captures only the essential aspects of the desired application or, in general terms, the SUT.

Model-based testing is a technique that reduces the amount of the user interaction by minimizing the manual tasks afforded by user, and producing tests automatically.

This technique only demand a well done model, that is indeed, the only part the user have to do; and the rest of procedure will be done by the tool itself automatically and basing-on this given model.

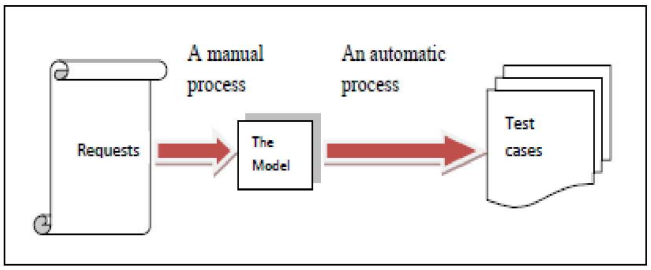

Figure 1. *Process of Model-based testing approach.*

4. Study of Existing Web Testing Tool

As a starting point of our project, a large study has been done while exploring the web testing tool's domain.

Many and various tools are provided; for that, users may wonder what tool to use when testing theirs applications. According to our research, we have selected a list of 20 web testing tools. The choice has been based on two major characteristics: At the first place, selected tools must be Java-based, afterwards, we have tried to figure out those between the large scale of Java based tools, what the most famous and most used are (this analysis has been concluded from the importance accorded to a given tool in professional forum discussions, web developers opinions, the total downloads number of the tool trial…).

Then, when actually trying to judge, in practical terms, the value of these pre-selected tools, some of them have been illuminated. Indeed, many problems occur when trying to evaluate some tools: Some of them display errors even before any action is performed (like cubic test), other ones seem to be inefficient or very complicated and incomprehensible for the regular user (like GUI Dancer and Jubula)…

4.1. Research Methodologies [4] [5]

In order to proceed in analyzing an existing phenomenon, a research became obligatory to obtain satisfactory results.

Consequently, many methodological ways of research have appeared to saturate this need.

Indeed, three main research methodologies coexist: Quantitative, qualitative and mix research.

Every methodology exists to cover a special part of the sample that the researcher wants to examine.

So, before beginning any research process, the researcher is demanded to determine first of all its analyzing goals.

According to our case, our goal is to analyze the qualitative part of existing web testing tools, in order to achieve a quality product. Underlying this final goal, our research orients immediately, to the qualitative methodology.

Speaking about this research category, we can mention that it is characterized by analyzing textual data that can be an interview, a conversational analysis, a questionnaire…and then the collection of all provided information to measure the sample's quantitative value.

Coming back to our study that consists in examining existing web testing tools, the questionnaire methodology seems to be the most suitable solution for this case. A well done questionnaire, can easily measures the sample functionalities, and starting from this perspective, we have prepared an inclusive eight-question-based questionnaire.

These methodological questions are posed in a way to study the global characteristics of a web testing tool: This study allows comparing between those tools, detailing their features and most importantly their limits…

4.1.1. Questionnaire for the Qualitative Research Methodology [6] [7] [8] [9]

Questions are held in the following part in the format of eight separated paragraphs

4.1.1.1. Testing Ajax Applications

The first question has as an aim to test if the tool is able or not to support Ajax functions: It is able or not to record normally Ajax-based tests? In fact, we have chosen to accomplish a small simple test, Ajax-based, and repeat this test with the entire selected tool list.

4.1.1.2. Multisession

Here the question is: Is the tool able to record a same test from two different web browsers? And does it link between them in a same action?

This means, if the chosen testing tool is able to record a test divided in two different web browsers, some actions of the test are done in a browser; others are done in the other one, to form a whole one test.

In this level we have chosen to figure out whether a tool is

able to record a chat action between two users from two different web browsers, in the same test. Accordingly, this test can be very useful to test web chatting pages, and it may be a critical measure to judge the tool.

4.1.1.3. Assertions

In this part we attempt to test if the tool is able or not to make assertions for HTML elements; we check if a user can be sure in a specific way that the desired HTML element exists or not in a specific web page.

Assertions are very important in the testing process. In fact, for some examples, the use of assertions is critical, like the case when a login process fails; the application returns a page indicating that the operation failed. This implies that no error has been detected. However, this is certainly an error condition. If the user has included additional pages dependent on successful login, an error would very probably occur later in one of these pages.

It is therefore, crucial to use assertions to detect such situations. Assertions help the tester to identify an error condition in the early scenario and to avoid analyzing incomprehensible errors, due to previous failures.

4.1.1.4. Report and Results

In this part, our goal is to explore the tool interface and see if it is able or not to offer a report or any type of result indications. Also, this is a very important point, because the report can ensure users from the test results: it passes or not; if not, what kind of errors occur, and how to fix them.

An efficient tool must be able to give the maximum details about a test especially when it fails, to guide users to the right way in order to get the satisfying results. Therefore, this point is a critical way to judge the tool efficiency.

4.1.1.5. Multi-Browser Support

The question here is: Is the tool able to works with more than just one web browser?

A multi-browser tool allows the user to test its web application in different platforms and that is very useful to examine how a browser reacts with this application in order to make it more efficient and convenient for all web browsers.

In some cases a web application can work with a web browser and fails with another one, so a multi-browser web testing tool can warn the user to this case, so that he can deal with this kind of problems.

4.1.1.6. Options of Extracting HTML Elements

The sixth question is: Does the tool provide the user with any option to pick how it searches and extracts the HTML elements when playing a test?

Methods of extracting HTML elements are many: by id, by label, by index…

Every tool has a default method to refer to a recorded HTML element. Indeed it is the manner how the tool recognizes a selected element to be able to re-call it again when playing the test. Some options are non methodological and fail re-finding the element frequently, so that we can pick this question as a measurement agent of the tool importance and efficiency.

4.1.1.7. Error Handling

At this level, our question is how the tool handles with an error when it happens? Once again tools differ by their reaction in front of the error occurrence, and that may represent a factor of distinction between them.

Users may prefer a tool that allows them to choose how to react when an error occurs, for some purposes related with the test nature.

Some of them would prefer, for example, to stop the test execution and try to fix the error; others may want to know how the application continues in this case, so they pick the continuation mode when finding an error.

4.1.1.8. Exporting and Importing Options

Right here, we want to see if the web testing tool is able or not to import and export the test into a specific programming language.

Exporting projects or tasks are in some cases, very important, because that allows users to handle with the exported file: explore it, edit it…without being obliged to re-open the tool every time.

Besides, it can help them to translate the test into another programming language that can be very easy to work with for some users.

4.1.2. Analyzing of the Chosen Web Testing Tools [10]

The questionnaire methodology consists in analyzing some given samples and making conclusions in order to improve and develop those samples. We have chosen to illustrate all the quantitative study done before, in a small table that aims to allow a direct and easy comparison between deferent tried tools.

In our project we choose to focus on what we have considered as the most important and common limits: the problem of use of the dynamic ID while recording actions and saving HTML objects during the test, and the un-multisession issue.

Consequently to these major limits, we can conclude that the “record and replay” technique is unreliable technique. As a matter of fact, it is the “record and replay approach” that we have tried with all previous tools, and we have discovered that it contains many deficiencies.

5. The Proposed Approach [1]

To avoid all the record and replay approach problems, we have chosen to create a model-based testing application.

The most important benefit of model-based testing is that instead of performing many test cases manually the tester can replace this big effort by just designing a model that describes the expected behavior of the application that he wants to test.

From the user requests from a given application, the user has to generate all test cases manually with the traditional approach, but the model-based testing technique saves the tester effort by generating all tests cases automatically; only the model has to be done manually.

From a given model, the user can select a specific algorithm that permits the test case generation, so the tester can obtain different test cases depending on the used algorithm.

Our project relies on this approach because of its diverse benefits. In fact, model-based testing technique:

- Reduces time costs: Many published studies show that this technique reduces time while testing. It is a fact that an abstract model can take time to be well designed but in case of deriving many test cases of the same application, the automatic test generation will be faster and easier than the manual one even when including the model designing time.
- Helps to detect logical errors: Besides its natural role, detecting the functionality errors, model-based testing can help to detect some requirement errors in the application structure and design, which means that while designing the model, the tester can detect some errors or lacks in the application conception and this gives him the possibility to rectify and develop the application after or even while testing the model.
- Is easy to be update: One important advantage is that when the application requirements change, it will be very difficult to be done once again manually: In case of large amounts of test cases, it will take time and patience to repeat all test cases manually, one per one. In case of model- based testing the user has only to update his model and re-generate new test cases automatically from the new updated model.

The main disadvantage of the approach is the time that has to be spent while modeling an abstract model and also the necessity of the expertise of some programming language and modeling skills to achieve a good model

Our application will be in a flash representation a graphical user interface GUI that is divided into two parts:

- General test scenario creation: Based on, like explained before, the model-based testing approach to automate test case generation.
- Specific test scenario creation: That permits the user to generate a specific test case manually.

So, like that, our project will be like a combination between the two approach advantages:

- The possibility to automate test generation, especially in case of large amounts of test cases are needed, to profit from the model-based testing technique and the test automation. This technique will be faster and easier in our project by simplifying the model designing compared to the existing one. This plus will make models easier to be modeled by the user and will reduce the amount of errors in the test. The simplification of the existing model format will be explained later on the conception part to explain the conception of the new model standard and then in the realization chapter to demonstrate how we have realized the designed model.
- The possibility to pick the manually recording test in case of a specific test case, because if the tester wants to test just one specific case, the automatic mode will generate many test cases, and that will demand a long time to find the suitable and desired case if it exists. So, in this case, a manual test will be a faster, easier and more practical solution.

The case of a specific scenario creation will be also model-based, despite that we will not use the model to generate test cases automatically, but it will be used to build a step by step test case, in terms of the user choice.

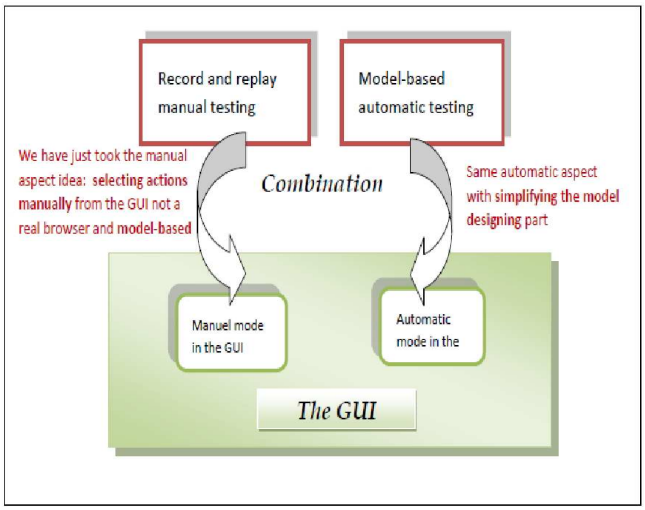

Figure 2. *The architecture of the GUI.*

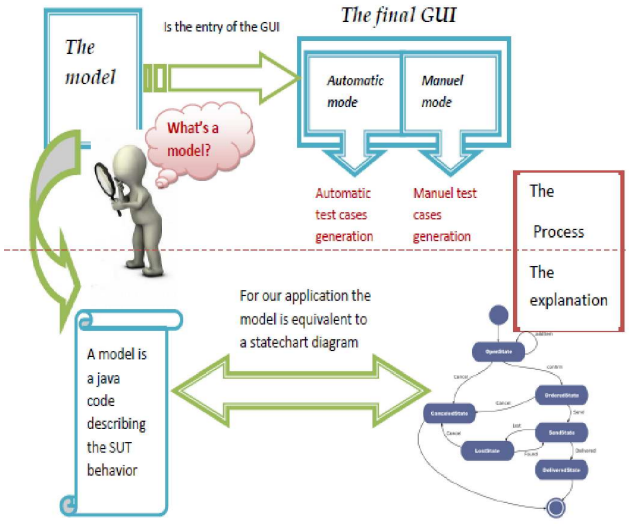

Figure 3. *General architecture of application.*

For web testing applications, models will be like a behavior's description of a given website. For that, the model can be considered as a group of states that represent all possible status of the application. Moving from a state to another is in fact a result of a specific action that leads the application from a status to another one.

So like that, a model can be regarded as a combination of states and actions that are related in a logical and sequential way to form a state chart diagram.

For more understanding the following figure can translate the internal components interaction in the application, relating the testing process to its logical explanation.

6. The Conception of the GUI [11]

6.1. Global Structure of the Application

The following figure recapitulates the application's behavior. Red boxes are the actions that have to be done by the user, and the green ones are those afforded by the application.

All user actions are afforded by simple mouse interactions with the tool. The only important action that the user has to do is the model designing.

After selecting the model describing the SUT behavior, the user can follow the previous steps indicated in the figure: First of all, the user has to select a testing mode through the first window of the application. Second, the GUI will automatically extract all the constitutions (states, actions and assertions) of the given model using the java reflection. If the selected mode is automatic, test cases generation will be done automatically after picking a specific generation algorithm (an algorithm is a specific manner that the application can follow to produce automatic testes). If the user choice is the manual mode, the tool will display all the possible actions in the model, and then the user can select a specific one between them.

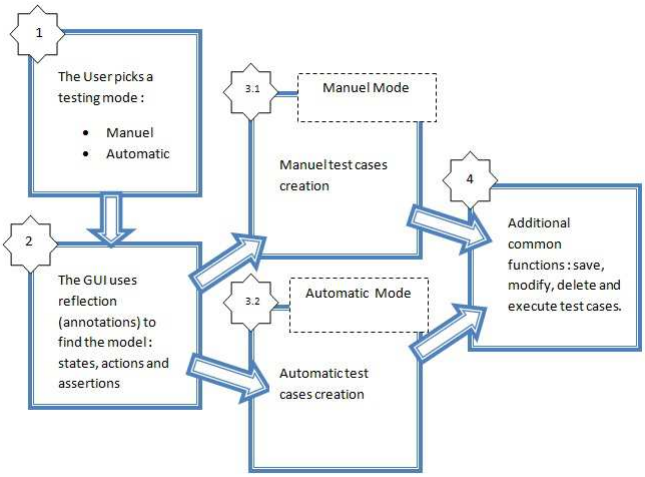

Figure 4. *Structure of the GUI.*

From this chosen action the tool will display the state that the action has led to. From the new state, the tool will display all the possible actions in this particular state so that the tester can pick another action leading to a new state and so one in order to form the desired test case (traveling in a state chart diagram). After creating automatic or manual test cases, the user can modify, save, execute or delete any test he wants.

7. Conclusion

In this paper we have tried to demonstrate the need of a new approach for testing web applications.

Previously while speaking about the adopted solution according to traditional testing ways, we have mentioned that we have improved existing models format, for the automatic testing mode. The old model has to handle with all the existing actions in the SUT independently, which means that the model designer has to produce all the possible actions and relate them to form a state chart diagram. This process can be very long and fatiguing when designing web application models, because they are complicated and rich. So, we have merged some related actions into one action. For example, to perform a login process to a specific website, the user has to accomplish three successive actions: entering the user pseudo name or email, then the password and finally clicking the submit button. By merging those three actions in a unique action, we can save the designer effort and time.

References

[1] D. Souilem Boumiza and A. Ben Azzouz and F. Ben Brahim 2012 "Design and development of a user interface to customize web testing scenarios" *International Conference on Education & E-Learning Innovations ICEELI' 2012 , July 1-3 Sousse,Tunisia*

[2] C. Eaton and A. M. Memon: "An Empirical Approach to Testing Web Applications Across Diverse Client Platform Configurations" by. International Journal on Web Engineering and Technology (IJWET), Special Issue on Empirical Studies in Web Engineering, vol. 3, no. 3, 2007, pp. 227–253, Inderscience Publishers.

[3] T.Banner, H.Eicher, A.Rennoch. "Model-based testing in practice". 2nd workshop on model-based testing. In practice MOTIP 2009.

[4] Sampath, R. Bryce, Gokulanand Viswanath, Vani Kandimalla, A. Gunes Koru: "Prioritizing User-Session-Based Test Cases for Web Applications Testing". Proceedings of the International Conference on Software Testing, Verification, and Validation (ICST), Lillehammer, Norway, April 2008.

[5] Research methodologies: concluded from http://www.enge.vt.edu", on april 2012.

[6] Testing anywhere: concluded from http://www . softwaretesting.net/ otherproducts/ testinganywhere .html , 2012.

[7] Win Task: concluded from http://www .csscody.com/ resources / web -testing- tool -list , 20013.

[8] Bad Boy: concluded from "http:/www.badbody.com.au/" 2012.

[9] Sahi: "http:/www.sahi.co.in/w/" 2012

[10] Fabasoft app.test : concludes from "htt://en.wikipedia.org/ wiki/fabasoft_app.test" , April 2012.

[11] Unified Modeling Language: Extract from "http://en.wikipedia.org/wiki/ Unified_Modeling_Language" , May 2012.

[12] Softwareengineering: http://en.wikipedia.org/wiki/Software _engineering, on April 2012

[13] J. Ernits, R. Roo, J. Jacky, M. Veanes «http://research.microsoft.com/pubs/101196/extended_version .pdf » On August 9, 2012

[14] B. Hayduk, « http://searchsoftwarequality.techtarget.com /tip/Model-based-testing-for-Java-and-Web-based-GUI-applic ations » On August 9, 2012