A Platform Independent Testing Tool for Automated Testing of Web Applications

December 10, 2009

Abstract

Increasing complexity of web applications and their dependency on numerous web technologies has made the process of testing web applications tedious, time-consuming and expensive. Many approaches have been proposed to automate the testing of web applications, where most of the approaches are confined to perform a specific type of testing. This proposal proposes an automated testing tool that detects the web technology used to implement the interface. Later it automatically selects an appropriate tool to test the web application based on the technology used, and the type of test to be performed, generates an automated test script, and executes it on the web application. Our approach is based on the concepts of machine learning where the developed tool is capable of automatically recognizing different technologies used by the web application and making intelligent decisions based on the given application. Unlike other existing approaches, this approach uses platform independent test cases and transforms them into platform specific tests suitable for the selected tool. In this way our approach supports the dynamic testing of any web application in a platform independent manner.

1 Introduction

Web applications play an important role in our day-to-day activities such as online banking, shopping and gaming. Also, many web applications are customized to cater to the needs of individual customers, which make use of new web technologies that provide them with a rich user experience, making the web applications handle more sensitive data and generate content dynamically. In fact, these sophisticated web applications contribute a fortune to the success of many well known companies. For this reason, quality assurance techniques aimed at web applications are important as companies want to be reliable and live up to the expectations of their customers.

A big problem is that the above mentioned properties add complexity to the web applications, which makes the process of web testing more challenging. Also, manual testing requires a considerable amount of human work making the testing process tedious and costly. Additionally, many techniques that work effectively when applied to simple web applications are not effective when applied to web applications that depend on different web technologies and may ultimately result in inadequate testing of the functionality of applications.

Recently, a considerable amount of work has been done in this direction, where models have been proposed to improve testing of web applications [4], and tools have been proposed to generate the test cases automatically for scripting languages based on user input [1]. But these tools require the tester to provide test scripts that are specific to a testing tool. This lack of support to a standard test script language poses a problem as different scripts would be required if the web technology used by the application is updated. Also, to perform a particular type of testing, different sets of tool-specific test scripts would be needed to test web applications developed with different web technologies.

The goal of this proposal is to build a tool that dynamically configures its test suite to automatically generate and execute platform dependent test scripts based on the technologies implemented by the web application to be tested. The major contribution of this proposal is an algorithm that:

- determines the web technology used by the web application that is under test
- automatically selects a tool from the given pool of tools
- automatically generates test scripts specific to the tool selected, and
- evaluates the results against the provided criteria

The tool implemented based on this algorithm would be quite useful to the society as the entire process of testing is completely automated and one single tool performs different types of testing on web applications developed on different platform which helps in saving a lot of time and effort in testing a web application.

2 Background

A web application is a software system that is accessed via a web browser over a network such as the Internet or intranet, which is composed of many modules that implement the applications functionality such as logging in, sending a request, processing a request etc. Every component of a web application has an entry point for the component called the root method which is called when the component executes. Web applications are developed over a client-server architecture where a client submits requests to the server component. When the request is received, the server invokes the root method, processes the request and sends the desired result back to the client. Common web applications include banking, online retail sales, webmail, etc.

Web applications gained popularity due to their rich functionality, simplicity to update and maintain the applications without distributing and installing software on multiple clients, and support for cross-platform compatibility. Client-side scripting is often used to create an interactive experience in the web applications. Technologies such as PHP, JSP, ASP, Ruby on Rails, etc. are used to coordinate client-side scripting with server-side applications. Web applications can be structured into logical chunks called tiers, where each tier has its own functionality. Usually, web applications are structured as 3-tiered applications as presentation, application and storage. A web browser is the tier-one (presentation); the engine, using some dynamic web content technology is tier-two (application); and the database is tier-three (storage).

Similar to desktop applications, web applications also have security risks and challenges to be countered. The functionality of the web applications is hampered due to various bugs or errors

present. Risks increase with the complexity of the application. So, testing of web applications is essential before they are used. Web application testing involves testing the performance of the applications, determining any errors present in the applications, checking for any vulnerabilities the application is suffering from, etc. Traditionally, manual testing was used which was tedious and time-consuming for complex applications. We focus on automated testing of these applications, where a manual process already in place that uses a formalized testing process is automated. Automating the testing process helps in improving the effectiveness, efficiency, and coverage of web application testing. Automated testing is advantageous over manual testing as it improves test accuracy, and saves time and cost of testing.

State of the Art.

The concept of automated testing of web applications was introduced in 2000 by Daniel and Boustedt [2], where they used a tool called web spider to automate the generation of test paths. In 2001, Ricca et al. [9] gave an overview on analysis and testing of web applications, where an UML model of web application was proposed to semi-automatically generate test cases. Sampath et al. [11] designed a framework to examine various web testing tools available in terms of scalability and effectiveness. Later in 2005, the same authors implemented and evaluated a set of automated replay techniques for user session-based testing of web applications [10].

Artzi et al. [1] proposed a method to dynamically generate tests to scripting languages and created a tool called Apollo, which used an oracle to determine whether the output of web applications is syntactically correct and automatically sorted, and minimized the inputs that expose failure. Wassermann et al. [15] proposed the method of concolic testing to automate test input generation for web applications, written in scripting languages such as PHP that will achieve better code coverage. These papers demonstrated the analysis of the methodologies to dynamically generate test inputs for web applications written in scripting languages.

There are many approaches for interface identification. Halfond and Orso [4] used a fully automated static analysis technique for discovering a web application interface, a set of input parameters and their potential values. They implemented their technique for JavaScript. They obtained better code coverage with test cases based on the interface extracted using their technique, as compared to the test cases based on the interface extracted using a conventional web crawler. More accurate interface identification can lead to a significant improvement in test-input generation and, this technique is more accurate than other techniques. The primary drawback of this approach is that the approximations it makes of an applications interfaces are very conservative. Subsequent work on interface identification by Halfond and Orso [6] addresses the limitations of their previous approach [4] by using a specialized form of symbolic execution. These techniques demonstrate the importance of precise interface identification to improve testing and analysis of web applications.

Halfond and Orso [3] proposed automated techniques for modeling web applications and used these models to improve testing and analysis of web applications, which are not just static HTML pages. The technique focused on more complex web applications that can dynamically generate HTML content, interact with external systems, and combine data from multiple sources. Halfond and Orso [5] presented an approach for automatically identifying parame-

ter mismatches in web applications, which deals with communication between two components of a web application. The flow of work in this field from the early 2000s, till the most recent times has been interesting, where research started with the analysis of various techniques present for testing of web applications, and the current research has focused on the design and implementation of new automated tools for a specific type of testing of the web applications.

Limitations Summarized.

Testing is a critical part of the software development process. A lot of different automated software testing tools have been proposed, but most of these tools perform a specific kind of testing and work with a specific language, for example, unit testing of a Java application. The automated tool Apollo proposed by Artzi [1] can be implemented only for PHP applications, hence, though it successfully tests the dynamically generated content, it cannot be used for applications using different languages. The approach proposed by Halfond [3] fails in identifying a precise interface to improve testing and analysis of web applications and also the approximations it makes of an application's interfaces are very conservative. In the later work of Halfond [3], he made use of the symbolic execution method, which makes the testing process more expensive. The approach specified by Di Lucca [8] does not generate an automated test script, rather the integrated platform used by him generates tests stubs and driver, which represents the test cases for a tool. And certain tools, such ReWeb and TestWeb, proposed by Ricca and Tonella, perform static analysis of web applications, rather than dynamic.

3 Challenges and Goals

This section presents challenges presented in the state of the art of the web application testing fields.

3.1 Challenges

There are three main open issues in this area:

Building a platform independent tool: Although there are a variety of tools to test web applications, most of them are either platform dependent or specific to a particular type of testing. This makes it difficult for the tester to choose a tool that is applicable to his web application. Therefore, we see the need for a tool that performs testing on the web application without considering the platform of the application and that performs multiple types of testing on a single application.

Using tool independent test scripts: Most of the testing tools require that the tester provide a test script that is specific to the testing tool, in order to perform testing. If we want to test (for the same functionality) the applications that use different technologies, then we have to use a different tool for each type of application and generate different test scripts that suit the need of that particular tool used. Therefore it is important to have a methodology to generate test scripts that are generic to the type of testing performed rather than generating test scripts that are specific to the tool.

A self managed system: Although there are several state of the art approaches that have automated the process of testing, they require a considerable amount of human interaction. The most important open issue in the field of software testing is to reduce the amount of human interaction with the testing tool.

3.2 Goals

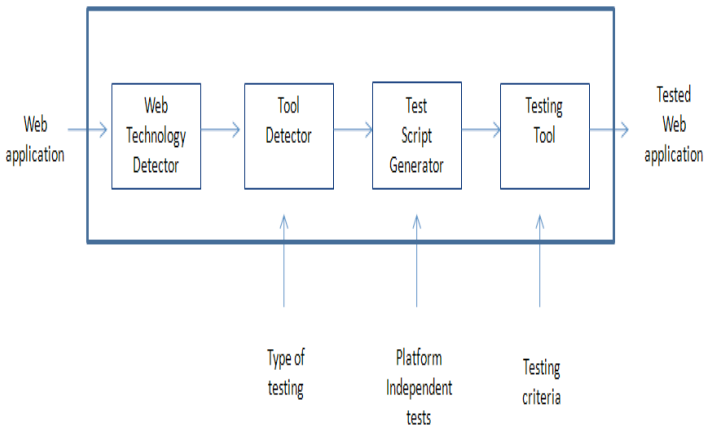
We have four goals to resolve the current issues in this research

1. To build a generic, platform independent tool for testing web applications
2. To build one single tool that can perform different types of testing on an application
3. To use tool independent test scripts
4. To reduce the amount of interaction between the tester and the testing tool

4 Proposed Research

From our investigation into the open research problems, we propose an automated, platform independent tool that performs a specific type of testing on the given web application. Our tool uses the concept of Machine Learning. The user is allowed to perform various types of testing such as Functionality Testing, Usability testing, Interface testing, Compatibility testing, Performance testing and, Security testing based on the requirements of his application.

Our approach examines the code of the web application and determines the type of technology used to implement the application using Machine Learning technique. The user is given an opportunity to request the type of testing to be performed on the web application. The detected web technology and the type of testing requested by the user are compared against the pool of automated testing tools, and an appropriate tool is selected for testing the web application. Based on the testing tool chosen, our tool generates appropriate test scripts and executes them on the chosen testing tool. The results obtained are evaluated against the predefined criteria.



Workflow Diagram of the Approach Used

4.1 Tool initialization

The most important goal of this proposal is to reduce human interaction with the testing tool. But, minimum initializations to the tool are necessary before using the tool on the web application. Platform independent tests have to be written which can later be transformed into test scripts to be used by a specific tool. The criteria for testing have to be specified, for example, code coverage. The type of testing to be performed on the application has to be specified, for example, interface testing, load testing, stress testing, compatibility testing etc.

4.2 Detecting the technology used

Our approach uses machine learning technique to detect the technology used by the given web application.

Machine Learning: Machine learning allows the system to change behavior based on data, where the system can automatically recognize

complex patterns and make intelligent decisions based on data. Machine learning has different types of algorithms such as supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, transduction, and learning to learn [13].

Machine learning in our approach: In our approach, a general inductive process, also called the textitlearner automatically builds a classifier for a given category (type of technology used) such as Java, PHP, Ruby on Rails etc. by observing the characteristics (such as specific tags used by the technology used to develop the web application, for example, an application using tags such as <html> , </html> will be categorized as HTML application) of a set of web applications that use different web technologies i.e., the textitlearner learns to intelligently classify a new unseen web application under the appropriate category based on its characteristics. This type of algorithm is known as supervised learning where the learning process is supervised by the knowledge of the categories and of the training instances that belong to them.

4.3 Test Script generation

Hernandez et al. [7] presented the abstract test scripting language that is independent of the testing tool used. Using the model-driven approach and the abstract test script language, tool specific test script can be generated.

Many test scripts are provided by the developer, where each test script defines the procedure to perform different types of testing such as Usability testing, Interface Testing, Compatibility Testing, Security Testing etc. All these test scripts are in abstract test script language which will later be converted into specific test scripts to suit the need of the chosen tool to perform testing on the web application.

5 Evaluation Plan

In order to evaluate the proposed tool we can consider web applications incorporated with different web technologies and perform various types of testing on these applications and the results of which help us deduce the effectiveness of this tool. The main research questions that has to be addressed in this evaluation phase is,

RQ1: Correctness: How effective is the tool in correctly identifying the web technologies used in the applications that are being tested?

RQ2: Reliability: How reliable are the test results produced by this tool, i.e., does the tool produce any false positives and what is the accuracy of these results?

To answer these questions, we will perform a case study, by testing two complex web applications involving different technologies, such as TenAday [14], a web application for conducting online examination, which is built using PHP and Scribd [12], a popular document sharing application which is built using Ruby.

Once the tool is run on these applications, it must rapidly configure itself to adapt to the structure of the application and should be able to effectively generate test scripts for a platform specific testing tool. The primary function of the tool is to detect and generate a report of the technologies implemented by the web application under test. Once this is identified, then we need to determine whether an appropriate test model is used in generating the appropriate platform specific test scripts. By this, we can evaluate how well we are able to implement supervised learning algorithm in developing our tool.

Finally, by validating the obtained results that

are logged by the tool, we will be able to address the reliability issues of the tool. We will then use the testing tool which was selected by our proposed tool from the pool, to test the web application independently, after which we can measure the correctness of our tool by comparing the results obtained from using these tools.

6 Summary of Foreseen Contributions

By using the proposed methodologies, a software tester can test web applications using different web technologies with a single tool. Since our tool performs different types of testing on a web application, the tester need not have the knowledge about different testing tools used for different types of testing. Also, the job of the testers is simplified further as they need not use different test scripts to test different applications, using different technologies. The most important contribution is all of these proposed methodologies are done automatically with minimal human intervention. This platform independent tool would be a great contribution to the society as the entire process of testing is completely automated and one single tool performs different types of testing on web applications developed on different platform which helps in saving a lot of time and effort in testing a web application.

References

[1] S. Artzi, A. Kiezun, J. Dolby, F. Tip, D. Dig, and A. Paradkar. Finding Bugs in Dynamic Web Applications. In *Proceedings of the 2008 International Symposium on Software Testing and Analysis*, 2008.

[2] L Daniel and J Boustedt. Automated testing of web application functionality. In *Masters thesis, University of Uppsala, Uppsala, Sweden*, 2000.

[3] William G. J. Halfond. Web application modeling for testing and analysis. In *Proceedings of the 2008 Foundations of Software Engineering Doctoral Symposium*, 2008.

[4] William G. J. Halfond and Alessandro Orso. Improving Test Case Generation for Web Applications Using Automated Interface Discovery. In *Proceedings of the Joint ESEC/SIGSOFT Symposium on the Foundations of Software Engineering*, 2007.

[5] William G. J. Halfond and Alessandro Orso. Automated identification of parameter mismatches in web applications. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, 2008.

[6] William G. J. Halfond and Alessandro Orso. Precise interface identification to improve testing and analysis of web applications. In *Proceedings of the eighteenth international symposium on Software testing and analysis*, 2009.

[7] Yanelis Hernandez, Tariq M. King, Jairo Pava, and Peter J. Clarke. A Meta-Model to Support Regression Testing of Web Applications. In *In SEKE*, 2008.

[8] G. A. D. Lucca, A. R. Fasolino, F. Faralli, and U. de Carlini. Testing of web applications. In *ICSM*, 2002.

[9] Filippo Ricca and Paolo Tonella. Analysis and testing of Web applications. In *Proceedings of the 23rd International Conference on Software Engineering*, 2001.

[10] Sreedevi Sampath, Valentin Mihaylov, Amie Souter, and Lori Pollock. A Scalable Approach to User-Session based Testing of Web Applications through Concept Analysis. In *Proceedings of the 19th IEEE international conference on Automated software engineering*, 2004.

[11] Sreedevi Sampath, Valentin Mihaylov, Amie Souter, and Lori Pollock. Composing a Framework to Automate Testing of Operational Web-Based Software. In *ICSM Proceedings of the 20th IEEE International Conference on Software Maintenance*, 2004.

[12] Scribd. using ruby on rails. In *http://www.scribd.com/*.

[13] Fabrizio Sebastiani. Machine learning in automated text categorization. In *ACM Computing Surveys (CSUR)*, 2002.

[14] TenAday. PHP web application. In *http://www.tenaday.co.in/*.

[15] Gary Wassermann, Dachuan Yu, Ajay Chander, Dinakar Dhurjati, and Hiroshi Inamura. Dynamic test input generation for web applications. In *Proceedings of the 2008 international symposium on Software testing and analysis*, 2008.