Enabling automatic testing of Modern Web Applications using Testing Plug-ins

M. Rajesh Babu,
M.Tech student Department of CSE,
JNTUA College of Engineering,
Anantapur,
Email id: rajeshbabu.rgm@gmail.com

Dr. S. Vasundra,
Professor,
Department of CSE,
JNTUA College of Engineering, Anantapur,
Email id: vasundaras@rediffmail.com

Abstract-Modern web applications are very dynamic in nature with rich user experience. Such applications typically use Web 2.0 and Asynchronous JavaScript and XML (AJAX) technologies. These applications are very different from conventional web applications as they use stateful C/S communication in an asynchronous fashion. The use agent is able to communicate with web server without explicit form submission. This capability makes the technology to support Rich Internet Applications (RIA). However testing such applications is a tedious task. This paper proposes a mechanism that automatically tests AJAX applications. The mechanism makes use of a crawler to capture the client side fields and generate a state-flow which is basis for the completion of automatic testing. A DOM tree is built based on the client side state which is useful to navigate to various parts of the tree in order to test the data. Different invariants of DOM tree are built for covering all states of the application. The proposed mechanism also makes use f plug-in concept for implementation of a tool that caters the present and future needs of the application. The practical results revealed that the proposed prototype is able to capture AJAX faults and report them to development team.

1. Introduction

When World Wide Web was first invented, the web applications were static in nature. There were only meant for sharing information that has been published. Later on many server side technologies and scripting languages came into existence in order to make the web applications dynamic and interactive. Java Applets also served to make the web applications popular as they too made them dynamic. The web applications bestow many advantages as they can be accessed from any corner of the world without geographical and time restrictions. More over the web applications eliminate the client side installations as they are browser-based applications. The modern web application development is based on Web 2.0 and AJAX [1] applications. The technologies make the applications to support the development of Rich Internet Applications (RIA). AJAX technology affected the way web applications are developed. There was a fundamental shift in the development process. More usability and more natural web application developer were made possible with AJAX [2]. However, the features come at a price as they are prone to errors due to their tasteful nature the supports event handling. The use of DOM at client side also makes it vulnerable [2]. There are many static testing techniques to improve AJAX applications. However, the static analysis proved to be inefficient as there are many dynamic dependencies when AJAX is used. Its interaction with web resources in web servers makes it so complex and dependent on runtime scenarios. For this reason testing an AJAX application is Challenging task. Recently developed tools like Selenium1 is capable of capturing state of AJAX application and thus test modern web applications. However, they also need lot of manual effort for testing such applications.

Therefore, this paper aims at developing a tool that can automatically test AJAX applications. The proposed mechanism automatically gets all invariants of states of the application and covers all possible tests. The proposed mechanism uses a crawler to reach user given values through UI elements presented in browser. The proposed mechanism uses multiple invariants of DOM tree or the state flow graph to obtain application specific state at any given point of time. We implemented a tool which is plug-in based. This tool has crawling infrastructure besides flexibility to add more plug-in in future for incremental functionality building of the tool. The main contribution of this paper is the tool which implements the mechanism based on plug-in and crawler. The proposed tool is capable testing RIAs for various kinds of inconsistencies. The automated testing of modern applications is possible with the proposed tool. The tool is influenced by [3] and [4]. The tool also gives comprehensive results that can help development teams to go ahead with fixing bugs if any. The remainder of this paper is organized as follows. Section 2 reviews literature on automated testing of modern web applications.

Section 3 provides details of the proposed automated testing mechanisms. Section 4 provides details of experiments and results while section 5 concludes the paper.

2. Prior Work

Modern web applications make use of plenty of client side scripting that interacts with server side technologies such as Java Server Pages (JSP), Active Server Pages (ASP), and PHP. As the technologies such as AJAX made the web applications more interactive and dynamic with rich user knowledge, they also incur errors in the applications. They are error prone. The literature on traditional and modern application testing is presented in this section.

A crawler was built in [5] to test web applications containing multiple pages. The crawler is capable of detecting inconsistencies such pertaining to page errors and navigation. The name of the crawler is VeriWeb. It can support testing web applications with client side scripting. However, it can’t cope with AJAX web applications. Web applications might have security vulnerabilities. To test such web applications there are some tools such as SecuBat [6] and WAVES [7]. The approach in these tools is to have a crawler to obtain data and detect abnormalities. They also detect malicious patterns pertaining to XSS and SQL with respect to various injection attacks. Model checking is used in [8] using a tool named [9] for testing applications built on web 1.0. In [10] a testing approach is followed which is model based. The tool implemented is ReWeb for creating web application using UML and a variety of test cases. Andrews et al. [11] presented another way that depends on state machine with constraints. All model based web application testing tools have a common drawback that is their inability to deal with AJAX applications [3]. For generating automated test cases in [12] and [13] the session data of logged in users is used. This kind of approach needs enough interaction with users of the web application. The session – based testing applications also focused on only synchronous requests that are synchronous in nature. The messages that come from server, popularly known as delta-server messages, are difficult to analyze. However, updates on such messages are more meaningful once they are processed at client side and injected into DOM. Static analysis concept is used to analyze server content. However it does not appear natural. Apollo is a tool implemented by Artzi et al. [14] which is to find faults in web applications made up of PHP. The tool is meant for detecting runtime errors that make HTML output inconsistent. In [15] and [16] presented an approach that analyzes server side Java code statically through request parameters to test their values. Symbolic execution of server side code is used in [17] for identifying possible interface of given web application. However, all such techniques have drawbacks. They can’t cope with complex client side behavior. Testing desktop applications was attempted by Memon in [18]. However, AJAX applications can be envisaged to be the combination of web applications and desktop applications [2]. To test such applications new tools are to be built that can cope with specific features of AJAX applications such as asynchronous c/s, dynamic etc. They are different from traditional interface applications. They make use of DOM – based interface that is not similar to desktop applications with Graphical User Interface (GUI) [19]. Any traditional testing tool can be used to test AJAX applications. JSUnit3 is one of the unit testing tools that can be used to test web applications that incorporate client side JavaScript. The mode widely used AJAX web application testing tools include Sahi, WebKing, and Selenium IDE. Other testing tools for web applications include Watij and Web Driver. These tools make use of APIs in order to control browser. However, they need lot of manual effort for testing web applications. In [20] Marchetto et al. discuss many traditional web testing techniques including session based testing [12], [13], model based testing [11] and code coverage testing [10]. These techniques work fine with web 1.0 application. When it comes to Web 2.0, they can’t be used. This paper proposes a tool that is plug-in-based to test modern web applications that make use of AJAX technology. This paper also focuses on crawler which is essential for automated testing of modern web applications.

3. Implementation of proposed Tool

The proposed tool is implemented to test modern web applications that make use of Web 2.0 and AJAX technologies. There are many challenges in testing such applications. They include finding methods that simulate user functionality in web application; gaining access to all states of DOM with respect to client; developing a method that can be used to find the correctness of the invariant states. In traditional web applications, the states are explicit in nature and can be tested easily [21]. However, testing the applications that use AJAX is not easy. This is because the event driven nature of AJAX is vulnerable. It may cause many addition attacks including SQL. In order to assert behavior of program, the proposed system makes use of invariants. In fact any dynamic application needs various DOM invariants and the testing has to consider all invariants of the DOM. Testing such invariants without human intervention are challenging. Prior works towards this end include Daikon [22] and DoDom [23]. The former takes invariants from runtime execution traces while the latter is also capable of obtaining DOM invariants in the client.

Obtaining AJAX States

We have built a crawler in Java programming language that can derive various states of an AJAX web application. This crawler was influenced the work in [3] and [24]. The crawling process has been implemented using two algorithms which are presented in figure 1 and 2.

```
1.  Procedure START(url, Set Tags)
2.  Browser ←
    initEmbeddedBrowser(url)
3.  Robot ← initRobot
4.  sm←initStatemachine()
5.  precrawlingPlugin(browser)
6.  crawl(null)
7.  postcrawlingPlugin(browser)
8.  end Procedure
9.  procedure CRAWL(STATE ps)
10. cs←sm.getCurrentState()
11. update←diff(ps.cs)
12. f←analyseForms(update)
13. set C←
    getCandidateClickables(update,t
    ags,f)
14. for c= do
15. Generate Event(cs,c)
16. End for
17. End procedure
```

Figure.1-Crawling Process

As can be seen in fig. 1, the algorithm has functionalities pertaining to crawling web pages including pre-crawling and post – crawling functionalities. While crawling the algorithm generates various events. The process of generating events is presented in figure. 2.

```
1.  Procedure Generate Event(State
    cs,Clikable c)
2.  Robot.enterFormValue(c)
3.  Robot.fireEvent(c)
4.  dom←browser.getDom()
5.  ifstatechanged(cs.getDom(),Dom)then
6.  xe←getXpathExpr(c)
7.  ns←sm.addState(Dom)
8.  sm.addEdge(cs,ns,Event(c,xe)
9.  sm.changeToStaePlugins(ns)
10. runOnNewState(ns)
11. test Invariant(ns)
12. ifstateAllowedToBeCrawled(ns) then
13. crawl(cs)
14. end if
15. sm.changeToState(cs)
16. ifbrowser.history.canGoBack then
17. browser.history.goBack()
18. else
19. {We have to back-track by going to
    initial state}
          20. Browser. reload()
21. List E←sm .getPathTo(cs)
22. For e=E do
23. re←resolve Element(re)
24. robot .enter Form Values(re)
25. robot .fire Event(re)
26. end for
27. end if
28. end if
29. end procedure
```

Figure. 2 –Event Driven Analysis of AJAX States

As can be seen in figure. 2, the generation of events is done robotically. A software robot moves into HTML forms and fields and generate appropriate events that are used to test the modern AJAX web applications automatically.

Plug-in Development As part of the tool two plug-ins were implemented. The first plug-in is meant for finding development errors. Such errors are due to wrong understanding of requirements and design. The second plug-in is meant for discovering security vulnerabilities. The development errors are pertaining to AJAX coding and also the responses from web resources that run in the web server. Such errors are identified based on the expected behavior of the web application. Navigation and other errors are identified and reported. With respect to security vulnerabilities, the proposed tool analyzes dynamic AJAX functionality by inspecting the code closely to know whether the code is intact or tampered to inject SQL and other injections. Even JavaScript code can be injected into a web application to make it compromise towards certain functionalities. This will help hackers or intruders to gain control over web pages. The plug-in checks the vulnerabilities and reports to development team. More plug-ins can be developed in future to make the application more robust and cover various kinds of errors.

4. EMPERICAL RESULTS

The proposed tool is tested with Web 2.0 applications that make use of AJAX. The tool is tested to know its various functionalities. First of all its ability to obtain meaningful invariants is tested. It has been observed that writing invariants needs good understanding of design; Xpath expressions can be used to represent invariants; invariants can be used to discover faults; with invariants, manual effort is greatly lessened. In another experiment fault revealing capabilities of the tool are tested. It has been observed that the tool is capable of reporting faults; provides acceptable performance with respect to crawling and also automatic testing of web applications. In the third experiment focus is on AJAX enabled applications. The results revealed that the tool is capable of finding faults with the application.

5. Conclusion

In this paper we have implemented a tool in Java which can test modern web applications that make use of AJAX technology. Towards this we have implemented a crawler which captures data given by the user in web browser. Then the tool makes use of DOM tree and invariants of the state flows in order to test the web application for discovering inconsistencies. Many fault models have been used in order to uncover hidden bugs in the AJAX applications. An algorithm is built for obtaining all state flows while crawling the web application. The application is modular and scalable in nature. The plug-ins concept makes it to increment its functionality in future with ease. In addition to basic level testing, this tool supports uncovering of development faults and also security vulnerabilities. The experimental results revealed that the tool is very useful in testing real time web applications that provide rich user experience.

References

[1] J.Garrett,“Ajax: A New Approach to Web Applications,”adaptivepath,http://www.adaptivepath.com/publications/essays/archives/000385.php, Feb. 2005.

[2] A. Mesbah and A. van Deursen, “A Component- and Push-Based Architectural Style for Ajax Applications,” J. Systems and Software,vol. 81, no. 12, pp. 2194-2209, 2008.

[3] A. Mesbah, E. Bozdag, and A. van Deursen, “Crawling Ajax by Inferring User Interface State Changes,” Proc. Eighth Int’l Conf.Web Eng., pp. 122-134, 2008.

[4] A. Mesbah and A. van Deursen, “Invariant-Based Automatic Testing of Ajax User Interfaces,” Proc. IEEE 31st Int’l Conf. SoftwareEng., pp. 210-220, 2009.

[5] M. Benedikt, J. Freire, and P. Godefroid, “VeriWeb: Automatically Testing Dynamic Web Sites,” Proc. 11th Int’l Conf. World Wide Web,pp. 654-668, 2002.

[6] S. Kals, E. Kirda, C. Kruegel, and N. Jovanovich, “Secubat: A Web Vulnerability Scanner,” Proc. 15th Int’l Conf. World Wide Web,pp. 247-256, 2006.

[7] Y.W. Huang, C.H. Tsai, T.P. Lin, S.K. Huang, D.T. Lee, and S.Y.Kuo, “A Testing Framework for Web Application Security Assessment,” J. Computer Networks, vol. 48, no. 5, pp. 739-761,2005.

[8] L. de Alfaro, “Model Checking the World Wide Web,” Proc. 13thInt’l Conf. Computer Aided Verification, pp. 337-349, 2001.

[9] L. de Alfaro, T.A. Hen zinger, and F.Y.C. Mang, “MCWEB: A Model-Checking Tool for Web Site Debugging,” Proc. World Wide Web Conf., posters, 2001.

[10] F. Ricca and P. Tonella, “Analysis and Testing of Web Applications,”Proc. 23rd Int’l Conf. Software Eng., pp. 25-34, 2001.

[11] A. Andrews, J. Offutt, and R. Alexander, “Testing Web Applications by Modeling with FSMs,” Software and Systems Modeling, vol. 4, no. 3, pp. 326-345, July 2005.

[12] S. Erlbaum, G. Rothermel, S. Karre, and M. Fisher II, “Leveraging User-Session Data to Support Web Application Testing,” IEEETrans. Software Eng., vol. 31, no. 3, pp. 187-202, Mar. 2005.

[13] S. Sprenkle, E. Gibson, S. Sampath, and L. Pollock, “Automated Replay and Failure Detection for Web Applications,” Proc. IEEE/ACM 20th Int’l Conf. Automated Software Eng., pp. 253-262, 2005.

[14] S. Artzi, A. Kie_zun, J. Dolby, F. Tip, D. Dig, A. Partaker, and M.D. Ernst, “Finding Bugs in Dynamic Web Applications,”Proc. Int’l Symp. Software Testing and Analysis, pp. 261-272,2008.

[15] W. Halfond and A. Orso, “Improving Test Case Generation for Web Applications Using Automated Interface Discovery,” Proc.Sixth Joint Meeting of the European Software Eng. Conf. and the ACMSIGSOFT Symp. the Foundations of Software Eng., pp. 145-154, 2007.

[16] W. Halfond and A. Orso, “Automated Identification of Parameter Mismatches in Web Applications,” Proc. 16th Verification and Validation, pp. 128-136, 2010.

[17] W. Halfond, S. Anand, and A. Orso, “Precise Interface Identification to Improve Testing and Analysis of Web Applications,” Proc.18th Int’l Symp. Software Testing and Analysis, pp. 285-296, 2009.

[18] A. Memon, “An Event-Flow Model of GUI-Based Applications for Testing: Research Articles,” Software Testing, Verification and Reliability, vol. 17, no. 3, pp. 137-157, 2007.

[19] A. Marchetto, P. Tonella, and F. Ricca, “State-Based Testing of Ajax Web Applications,” Proc. IEEE First Int’l Conf. Software Testing Verification and Validation, pp. 121-130, 2008.

[20] A. Marchetto, F. Ricca, and P. Tonella, “A Case Study-Based Comparison of Web Testing Techniques Applied to Ajax Web Applications,” Int’l J. Software Tools for Technology Transfer, vol. 10,no. 6, pp. 477-492, 2008.

[21] A. Mesbah and A. van Deursen, “Migrating Multi-Page Web Applications to Single-Page Ajax Interfaces,” Proc. 11th Europea n Conf. Software Maintenance and Reeng. pp. 181-190, 2007.

[22] M.D. Ernst, J. Cockrell, W.G. Griswold, and D. Notkin, “Dynamically Discovering Likely Program Invariants to Support Program Evolution,” IEEE Trans. Software Eng., vol. 27, no. 2, pp. 99-123,Feb. 2001.