IJDCST @October Issue- V-1, I-7, SW-5
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

Dynamic State Testing of Web 2.0 Applications

[1]K V NAGARJUNA KRISHNA, [2]A.RAMANA LAKSHMI
[1] Student, DEPT.OF CSE, PVPSIT, KANURU, VIJAYAWADA.
[2] Associate. Professor, DEPT.OF CSE, PVPSIT, KANURU, VIJAYAWADA.

Abstract: AJAX applications are notoriously error-prone due to, e.g., their state full, asynchronous, and event-based nature, the use of (loosely typed) JAVASCRIPT, the client-side requirement of the browser's Document-Object Model (DOM), and the use of delta communication between client and web-server. Unfortunately, static code analysis techniques are not able to reveal many of the dynamic dependencies present in today's web applications. To that end, we propose an implementation named ATUSA in which we automatically derive a model of the user interface states of an AJAX application by "crawling" the AJAX application code to identify key events and components. In order to recognize logical failures in these executions, we propose the use of invariants obtained from the crawling process and for handling structural failures we propose to implement Metamorphic Relations based oracle. The results highlight the efficiency of the proposed approach in terms of fault-detecting performances, reliability and scalability, automatic construction model and the usefulness of invariants and metamorphic relations.

Index Terms: Ajax, JavaScript, Testing and Debugging, Metamorphic Testing, Machine Learning, Metamorphic relations, Automatic testing.

I. INTRODUCION

Now a day's growing trend to development of modern applications through web is the main aspect for web testing. A well known examples include the Google's Gmail, word processing and Calendar applications. For this application development JSP and servelts technologies are used. But these technologies are used for accessing more updated information in web application development. For today's web applications a most related technology were introduced for development of above mentioned applications i.e. AJAX is the acronym for the Asynchronous JAVASRCIPT and XML. Use of AJAX technology positively affects the user friendliness and interactiveness of web applications. AJAX applications are noticed their stateful, asynchronous and event based nature for browser's Document Object Model between client browser and web server. In order to development of dependability of the AJAX applications and static analysis can be developed. This development is based on page request and response model. AJAX applications are commonly error prone due to their natularity and client manipulation of the browsers Document Object Model and use of communication between client and server. For this application development recently used tool such as Selenium offer a capture-and-replay style of testing for modern web applications. While such tools are capable of executing AJAX test cases, they still demand a substantial amount of manual effort from the tester. Then traditional tool was taking manual effort for develop these type of dynamic web applications. Existing tool can't be support for dynamic web testing. For dynamic user interface of AJAX applications we obtain a model by crawling AJAX applications to identify the key events and components present in AJAX applications. In this application Invariants are the properties of either client side DOM tree hold the execution.

Figure 1: Example for DOM tree construction.

As shown in the widget is a sub tree of the DOM object. Route element s having ids like widget#1 and widget#2. DOM tree was used to detect the violations in HTTP request and response present in the web applications. Including DOM violations in HTTP it consists crawling infrastructure similar to CRAWJAX as well as series of testing specific extensions; it refers ATUSA (Automatic Testing of User Specific AJAX). We have implemented these tools as a series of AJAX applications. Furthermore this testing tool was used to test the invariants present in the JAVASCRIPT modern applications. For this process we propose to extend our ATUSA tool for supporting JAVASCRIPT modern applications. Only the programmer can understand if the analyzer points to a real error or it is just a false positive. The necessity to review false positives takes work time and weakens attention to those code fragments that really contain errors. In this paper we propose to implement metamorphic testing. Metamorphic testing is a technique for the verification of software output without a testing complete testing oracle. The subject program is verified through metamorphic relations (MR). It is unlikely for a single MR to detect all possible faults. Therefore, four MRs that are quite different from one another with a view to detecting various faults will be used. Finding good MRs requires knowledge of the problem domain, understanding of user requirements, as well as some creativity.

II. LITERATURE REVIEW

As mention in the above discussion modern web interfaces in client side user manipulation is separated with server side application logic developed by the programmer. This testing process considers two types of web testing's; firstly introduce traditional web testing and graphical user interface web testing in developing of web applications. In traditional web testing doesn't provide sufficient details to determine the weather cope with modern AJAX applications. In this system testing tool is based on the crawler capable of detecting data entry points. Traditionally ReWeb, a tool was creating a model of the application. Logging of the user session present data on the server side. Is used for automatic testing. Above techniques have limitation on solving faults that are due to the complex runtime of the web applications. Due to dynamic nature of the AJAX applications with specific nature features such as client side communications and asynchronous which may have graphical oriented applications. For these graphical AJAX applications with invariants present in the applications currently Selenium IDE, for testing applications. Detection of invariants is another direction for modern web applications, Diakon a tool capabiling of a exploring ways of automatic detection of invariants in both AJAX and JAVASCRIPT web applications.

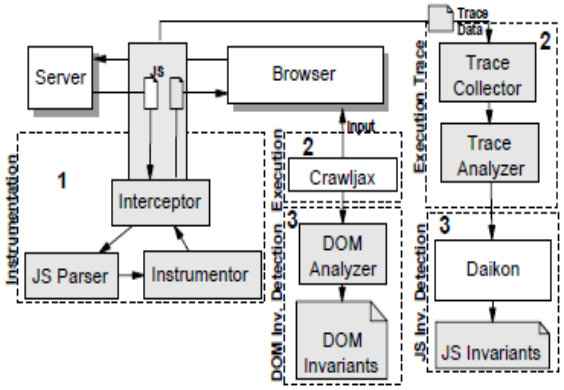

Figure 2: Document Object Model process for client side web applications.

As shown in the above client side applications for generating test cases using document object model. In this model we can access the efficient test case results based their invariants or properties present in both AJAX and JAVASCRIPT web applications. But for most generation of test cases above mention AJAX and JAVASCRIPT based technologies can be worked efficiently by using metamorphic testing present in latest testing facilities. In metamorphic testing we are using some metamorphic relations like functions based regular expressions present in the mathematical calculation. Our proposed work apply in these achievements efficiently when compare to ATUSA on both AJAX and JAVA SCRIPT web applications.

III. BACKGROUND WORK

Today's dynamic nature of the web applications, nature can be calculated in two ways static analysis and dynamic analysis. Static analysis is the analysis of the software applications in without executing programs, dynamic analysis is the analysis of the computer software applications with execution of programs in sequential order. Due to supporting of this nature for dynamic web applications one of the deriving technology like AJAX acronym for Asynchronous JAVASCRIPT and XML. In this applications user navigation is not necessary for accessing services through HTML tags. Due to the asynchronous nature of the AJAX applications and JAVA SCRIPT the client side manipulation of the web browser's Document Object Model and delta communication between client and server. For this communication between client and server testing process traditionally developed testing tool like Selenium, it captures and replay style of testing modern web applications. But In this tool also contains manual effort on user for generating test case generations. For automatic test generation of testing of web applications we obtain a model for testing AJAX applications. We automatically derive a model of the user interface states AJAX applications. We obtain above model by "crawling" an AJAX application code to identify key events and components. For developing invariants of the client side Document Object Model in execution state we are developing crawling infrastructure similar to CRAWLJAX as well as series of testing specifics (Invariants: properties of the either client side DOM tree derived state machine that should hold for any execution) referred to as ATUSA acronym for Automated testing of User Specific AJAX. The results highlight the efficiency of the existing approach in terms of scalability and automation model for testing approach.

IV. PROPOSED WORK

Automatically detecting dynamic structural and JAVASCRIPT invariants in modern web applications is complicated due to short comings of Static code analysis techniques. Static code analysis has two disadvantages; firstly Static analysis is the normally poor regarding diagnosing faults and concurrency of the errors. For detecting these type of errors we actually necessary for executing a part of the program virtually. It is difficult to implement because such programs takes more space for processing them. Dynamic analysis is the most efficient tool for detecting concurrency errors present in program. Secondly false positive reports, these type of errors can understand only by the programmer. False positive takes work time and weakens attention to those code fragments that really contain faults. Above mention static code analysis based on oracles for initiating testing. We propose to extend ATUSA in metamorphic testing. Metamorphic testing is a technique for the verification software application without complete of oracle testing. Metamorphic testing observes executions do not result in failure, they still provide useful information. This testing can be used for construction test cases from original set of test cases with reference to selected necessary properties of the specified function. Such functions are referred to as metamorphic relations. A metamorphic relation is used for detecting all possible errors in program. A good metamorphic relation is it requires knowledge of the problem domain with understanding of the user requirements as well as some creativity. So this kind of testing facilitates in an automated addressing of all possible forms of failures be it structurally or logically.

V. METAMORPHIC RELATIONS

We outline the MRs that we have a tendency to anticipate classification algorithms to exhibit and additional functions as follows:

MR-0: Consistence with cosine transformation. The result can be the same if we apply the related arbitrary cosine transformation function, $f(x) = ax + b$, $(ax= 0)$ to every value x to any subset(S) with training data set S and the test cases.

MR-1.1: Permutation of the prescribed class labels. If source result is ve then apply prescribed class labels followed up case.

MR-1.2: Permutation of the property aspects. If permute the n attributes of all samples and therefore the test data.

MR-2.1: It describes the uninformative with their properties. Selection of the actual value to be added here is necessary as this attribute with related class labels.

MR-2.2: Addition of informative attributes. According to the source input then results were obtained and specify the attributes powerful related class with different classes.

Metamorphic set contains programs logic, implementation of metamorphic relations with evaluation results. We are taking testing results of the every method present in the web application. Follow-up test cases ought to be created from the original set of test cases with relation to designated necessary properties of the desired functions.

Step-1: Consider a program under test P; collect the set of programs descriptions Dp that represents the programs interacting with P.
Step-2: Design a metamorphic relations MR_i applicable to test P.
Step-3: Implement MRi in the metamorphic set MS of the P.
Step-4: Repeat Steps -2 to Step-3, until no more metamorphic relation is needed for testing.
Step-5: For each available successful test case t_o, do
  i. MS uses applicable MR_i to construct the following-up test case t_f of t_o.
  ii. MS invokes P to execute t_f.
  iii. MS obtains the final results t_f
  iv. If MS detect a failure by using MR_i, then report the failure and go to Step (step-7).
  v. Repeat Steps-5(i) to step-5(iv), until no more applicable MR_i.
Step-6: Report that no failure is found.
Step-7: Exit

Figure 3: Algorithm for working of metamorphic relations.

In Step-1, collects the program description that the program under test. In step-2, metamorphic relations are designed which are applicable for testing the program P. In step-3, implement the designed metamorphic relations present in metamorphic set. The above two steps i.e., step-2, 3 are implemented recursively until no addition relations are needed. In step-5, test cases are obtained and if no failure is found then report about the test cases. If failure found then exit, and re-apply the metamorphic relations.

VI. PERFORMANCE ANALYSIS

In this section we describe the results of testing results mention in the above discussion. Load the program for ATUSA tool for testing application working procedure. Invariants are the properties and events present in the program. We define invariants should be hold and checked on the dynamic states with specific AJAX and JAVASCRIPT application development. We currently support for invariants in XPath regularity with efforts.

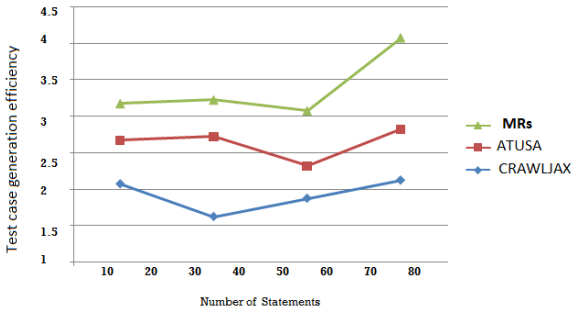

Figure 4: Comparison results with ATUSA and metamorphic relations.

Meta metamorphic relation is used for detecting all possible errors in program. As shown in the above figure testing performance results can be calculated with each function present in the metamorphic relation. For example we taking input as web applications then our proposed metamorphic relations can be applied on existing source code present in the

web application. Then our proposed work can be give results according their source code have equivalent testing feasibilities or not. We are applying each metamorphic relation present in the metamorphic testing using regular expressions. For example f(x) = ax+b is an regular expression for solving coordinate feasibilities in mathematical calculation. By using these type of relations present in the metamorphic testing feasibilities are gathered accurately.

VII. CONCLUSION

Unfortunately, static code analysis techniques are not able to reveal many of the dynamic dependencies present in today's web applications. To that end, we propose an implementation named MR driven ATUSA in which we automatically derive a model of the user interface states of an AJAX application by "crawling" the AJAX application code to identify key events and components. In order to recognize logical failures in these executions, we propose the use of invariants obtained from the crawling process and for handling structural failures we propose to implement Metamorphic Relations based oracle. These metamorphic relations are processed with equivalence and non equivalence relations among regular expressions.

VIII. REFERENCES

[1] "Automatic Invariant Detection in Dynamic Web Applications "Frank, Mesbah, and Arie van Deursen, ACM X-XXXXX-XX-X/XX/XX. ..$10.00 Copyright 2010.

[2] " Invariant- Based Automatic Testing of Modern Web Applications", Ali, Arie van Deursen, January/February 2012, IEEE Transactions On Software Engineering, Vol. 38, No. 1

[3] " A Framework for Automated Testing of JavaScript Web oriented Applications", Artzi, Julian, ACM 978-1-4503-0445-0/11/05 ...$10.00, *ICSE* '11, May 21–28, 2011, Honolulu, Hawaii, USA Copyright 2011

[4] M. Bened, J. Freire, and P. Froid, " Automatically Testing Dynamic Web Sites," pp. 654-668, 2002 Proc. 11th Int'l Conf. World Wide Web.

[5] "Finding bugs in dynamic web applications" J. Dolby, S. Artzi, A. M. Paradkar, and M. D. Ernst.ISSTA '08, July 2008, In Proc. Int. Symp. on Software Testing and Analysis.

[6] A. Mesbah, and A. Deursen, "Regression Testing Ajax Applications: Coping with Dynamism," Proc. Third Int'l Conf. 128-136, 2010, Software Testing, Verification and Validation, pp.