International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Fault Localization for Client Side Scripting of Web Application

Swati B. Ghawate[1], S. M. Shinde[2]

[1]PG Student, Computer Engineering Department, JSPM's JSCOE, Savitribai Phule University of Pune

[2]Associate Professor, Computer Engineering Department, JSPM's JSCOE, Savitribai Phule University of Pune

Abstract: ***Fault localization techniques are of more interest in recent years. These techniques are based on statistical analysis of program constructs executed by passing and failing test case executions. Till now Fault Localization techniques are available to locate fault automatically in server side dynamic web application. Existing system concentrate on fault localization of web application written in PHP. It shows how the Tarantula, Ochiai, and Jaccard fault localization algorithms can localize faults effectively in web applications written in PHP. It does not localize client side scripting faults, which plays a very important role in modern web applications . Fault localization of client side scripting is currently a tedious and mainly manual task. It is dynamic in nature and asynchronous which make it more error prone and challenging to debug. Our proposed approach is implemented with the help of open source tool Autoflox ,it present automated technique to localize client side scripting faults based on dynamic analysis of the web application.***

Keywords: Client side scripting ; Fault localization ; JAVASCRIPT ; AutoFlox ; JSP ;

1. Introduction

Background

Software debugging is the costliest and more time consuming process in software development. Also finding root cause of a failure is the most difficult process in debugging. So, techniques to automatically localize fault in software have come up.

Debugging can be divided into two main parts. The initial part is to recognize harmful code by using available testing technique. The next part is for programmers to actually examine the identified code to decide whether it certainly contains bugs. In the first part harmful code is prioritized based on its probability of containing bugs. The next part, assume that bug detection is perfect and it will direct programmers to the locate faults with least human involvement.

Web applications are written using combination of several languages, such as PHP/JSP on server side with AJAX and JAVASCRIPT at client. These applications generate output in the form of dynamically generated HTML. Automatic fault localization can help more to developer to test and locate fault of application. Our approach is designing fault localization model for client side scripting of web application to automatically locate faults.

Relevance

Understanding the nature of client side scripting in web application:
Client-side scripting empowers achieving dynamic and responsive web interfaces in web applications. It provides rich and dynamic user interface for web application. JAVASCRIPT, is used for client side scripting of application . It is dynamic, asynchronous and used to interact with DOM (Document Object Model) at run time. All these characteristics of JAVASCRIPT make it error prone and difficult to debug.

Challenges in Testing of Client Side Scripting:

In traditional programming languages, the goal of fault localization is to find the faulty lines of code. Due to JAVASCRIPT's asynchronous and dynamic characteristic some additional challenges are faced. The programmer will not only need to find the Page Layout faulty lines, but also have to map each executed sequence to the event that triggered their execution in order to understand the root cause of the error. In addition, event handlers may overlap, as a particular piece of JAVASCRIPT code may be used by multiple event handlers. Thus, manual fault localization in client-side JAVASCRIPT is a tedious process, especially when many events are triggered.

Debugging and locating fault of web applications is very expensive and mostly manual task. When the developers observe an error in a web program either spotted manually or through automated testing techniques , the fault-localization process get started. The developers then try to understand the root cause of the error by looking at the JAVASCRIPT code [5], examining the Document Object Model (DOM) tree, running the application again, and manually going through the initial series of navigational actions that led to the faulty state or running the corresponding test case. Manually isolating a JAVASCRIPT error's root cause requires considerable time and effort on the part of the developer. Thus, an error may propagate undetected in the application for a long time before finally triggering an exception. Further, errors may arise due to subtle asynchronous and dynamic interactions at runtime between the JAVASCRIPT code and the DOM tree, which make it challenging to understand their root causes. Finally, errors may arise in third-party code (e.g., libraries, widgets, advertisements), and

may be outside the expertise of the web application's developer.

Although JAVASCRIPT is syntactically similar to languages such as Java and C++, it differs from them in two important ways, which makes fault localization challenging.

Asynchronous Execution: JAVASCRIPT code is executed asynchronously, and is triggered by the occurrence of user-triggered events (e.g., click, mouseover), load events, or events resulting from asynchronous function calls. These events may occur in different orders; although JAVASCRIPT follows a sequential execution model, it does not provide deterministic ordering.

DOM Interactions: In a web application, JAVASCRIPT code frequently interacts with the DOM, which characterizes the dynamic HTML structure and elements present in the web page. As a result, the origin of a JAVASCRIPT fault is not limited to the JAVASCRIPT code; the JAVASCRIPT fault may also result from an error in the DOM. With regards to fault localization, the notion of an "erroneous line" of code may not apply to JAVASCRIPT because it is possible that the error is in the DOM rather than the code. This is particularly true for DOM-related JAVASCRIPT faults (described in further detail in Section 3), which are defined as JAVASCRIPT faults that lead to either exceptions or incorrect DOM element outputs as a result of a DOM access or update. As a result, for such faults, one needs to formulate the goal of fault localization to isolate the first line of JAVASCRIPT code containing a call to a DOM access function (e.g., getAttribute(), getElementById()) or a DOM update function/property (e.g., setAttribute(), innerHTML) that directly causes JAVASCRIPT code to throw an exception, or to update a DOM element incorrectly. This line is referred to as the direct DOM interaction.s

The Proposed System

Although fault localization in general has been an active research topic, automatically localizing web faults has received very limited attention from the research community. Automated fault localization for JAVASCRIPT-based web applications are not completely addressed yet. To make manual fault localization process easier for client side scripting of web application, in this paper, we propose an automated technique based on dynamic backward tracing of the web application to localize JAVASCRIPT faults. Our fault localization approach is based on Autoflox and Invarscope plugin for client side Javascript faults.
The main contributions of this paper include:

- A discussion of the challenges surrounding JAVASCRIPT fault localization, highlighting the real-world relevance of the problem and identifying DOM-related JAVASCRIPT errors as an important sub-class of problems in this space;
- A fully automated technique for localizing DOM-related JAVASCRIPT faults, based on dynamic analysis and backward tracing of JAVASCRIPT code;
- Implementing the fault localization technique with help of crwaljax , Autoflox and Invarscope plugin.

2. Related Work

All We classify related work into two broad categories: Fault localization of server side scripting and automatic testing of Client side scripting for web application.

Fault Localization for server side scripting of web application

The Early work on fault localization relied on the use of program slicing [12]. Lyle and Weiser [13] introduce, program dicing, a method for combining the information of different program slices. Renieris and Reiss [11] use set-union, set-intersection, and nearest neighbor methods for fault localization; these all work by comparing execution traces of passing and failing program runs. Shay Artzi, Julian Dolby [1] introduce, Tarantula, Ochiai, and Jaccard are different similarity coefficients that have been used for fault localization, by computing for each program construct a suspiciousness rating that reflects the likelihood for that construct to contribute to a failure. This suspiciousness rating of a program construct is computed as a function of the ratio of passing and failing executions that exercise it.

There have been numerous studies to find errors and vulnerabilities in web applications through static analysis. JAVASCRIPT is a difficult language to analyze statically [7], these techniques typically restrict themselves to a safe subset of the language. Automated testing of JAVASCRIPT based web applications is an active area of research [5], S. Artzi, J. Dolby designed a framework to automatically test JAVASCRIPT based web application. ATUSA [14] is an automated technique for enumerating the state space of a JAVASCRIPT-based web application and finding errors or invariant violations specified by the programmer. DODOM [6] dynamically derive invariants for the JAVASCRIPT code and the DOM respectively. However, none of these techniques focus on fault localization. Tools such as Firefox's Firebug plug-in exist to help JAVASCRIPT programmers debug their code. However, such tools are useful only for the bug identification phase of the debugging process, and not the fault localization phase.

Fault Localization for client side scripting of web application

Only paper that has explored fault localization in the context of web applications is by Artzi et al. [1]. However, their work differs from ours in various aspects: (1) they focus on the server-side code i.e. PHP, while we focus on the client-side; (2) they localize HTML validation errors, while our approach localizes JAVASCRIPT faults; and (3) they have opted for a spectrum-based approach based on Tarantula, while ours is a based on Tversky coefficient measure technique . Fault localization techniques isolate the root cause of a fault based on the dynamic execution of the application. They can be classified into Spectrum-based and Slicing-based. Spectrum-based fault localization techniques include Pinpoint , Tarantula and Whither. However, spectrum-based techniques are difficult to adapt to web applications, as web applications are rarely deterministic, and hence they may incur false

positives. Also, it is not straightforward to classify a web application's execution as success or failure, as the results depend on its usage. Our technique is similar to this body of work in that we also extract the dynamic backward slice of the JAVASCRIPT statement that throws an exception.

3. Implementation

Overview of the system

Our System attempt to locate Client side faults based on analysis of Java Script code under test. We are using following tools for the fault localization purpose.

1. AutoFlox.
2. Crawljax.
3. Invar Scope Plugin which needs to modify to satisfy project requirements .

Block Diagram

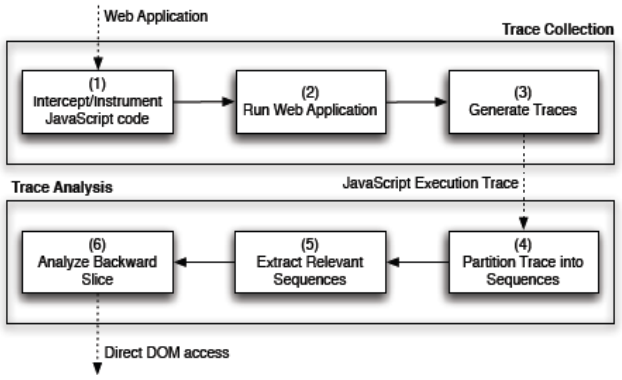

Figure 1: Block diagram consisting all phases of system

Our client side fault localizer consist of two phases:

- Trace Collection
- Trace Analysis

The trace collection phase involves crawling the web application and gathering traces of executed JAVASCRIPT statements until the occurrence of the error that halts the execution. In addition to the trace entries corresponding to the executed lines of JAVASCRIPT code, three special markers, called FAILURE, ASYNCCALL and ASYNC, are added to the trace. The FAILURE marker is used in the trace analysis phase to determine at which line of JAVASCRIPT code the exception was thrown; if a line l is marked with the FAILURE marker, then the value l:failure is set to true. The ASYNCCALL and ASYNC markers address the asynchronous nature of JAVASCRIPT execution . After the traces are collected, they are parsed in the trace analysis phase to find the direct DOM access.

Trace analysis phase begins after trace of executed statement is collected. The goal of this phase is to analyse the trace entries and find the direct DOM access responsible for the JAVASCRIPT failure. First, the approach partitions the trace into sequences, where a sequence (l1; l2; :::; ln) represents the series of JAVASCRIPT statements l1; l2; :::; ln that were triggered by the same event (e.g., a page load). Each sequence corresponds to exactly one event. After partitioning the trace into sequences, the algorithm looks for the sequence that contains the direct DOM access. This is called the relevant sequence. The relevant sequence is the sequence that contains the FAILURE marker. Once the relevant sequence has been found, the algorithm starts locating the direct DOM access within that sequence. To do so, it analyzes the backward slice of the variable in the line marked with the FAILURE marker.

4. Result

Our fault localization approach consists of two sections : (1) trace collection, and (2) trace analysis. The trace collection phase involves crawling the web application and gathering traces of executed JAVASCRIPT statements until the occurrence of the error that halts the execution. After the traces are collected, they are parsed in the trace analysis phase to find the direct DOM access.

The output of our client side scripting approach will be the direct DOM access corresponding to the error to be localized and has following aspects

1) The function containing the direct DOM access.
2) The line umber of the direct DOM access relative to this function,
3) The JAVA SCRIPT file containing the direct DOM access.

Following table shows the result of testing one small web application :

Table I

Javascript Error Type	Total Bug Reports	% of Total
Type 1	12	54.6
Type 2	6	27.3
Type 3	4	18.2
Total	22	100.1

Where

Type 1 - Code-terminating DOM-related
Type 2 - Output DOM-related
Type 3 - Unknown DOM-related

Result of the study on bug reports from testing web application and website:

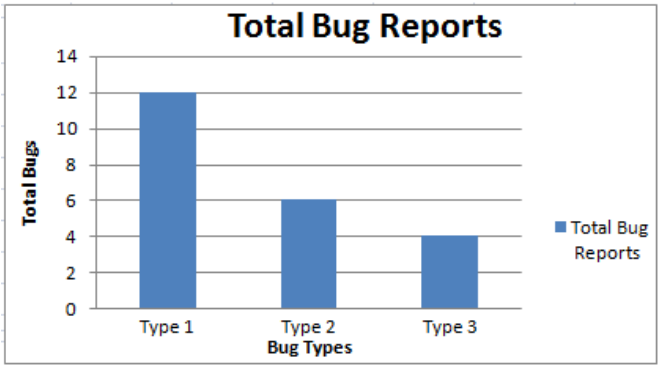

Figure 2: A sample line graph using colors which contrast well both Bug type and Total bugs

5. Conclusion

Till now Fault Localization techniques are available to automatically locate fault in server side dynamic web application. In this paper, we introduce a fault-localization approach for client side scripting of web applications. Our approach is based on trace collection and trace analysis of web application. It addresses the two main problems of client side scripting that are asynchronous execution and DOM interactions. We focus on JAVASCRIPT errors as we find that about 80% of the JAVASCRIPT errors reported in online bug databases of web applications pertain to the DOM. We have implemented our approach as an automated tool to localize faults of complete web application based client side script.

The current work focuses on code-terminating JAVASCRIPT faults, i.e., faults that lead to an exception thrown by the web application. However, not all DOM-related faults belong to this category. The design will therefore be extended to include a more automated technique for localizing non-code terminating JAVASCRIPT faults. This is also an avenue for future work.

6. Acknowledgement

It is with deep sense of gratitude that authors acknowledge the sincere help of concerned people for providing very constructive suggestions to improve the method.

References

[1] Shay Artzi, Julian Dolby, Frank Tip, and Marco Pistoia " Fault Localization For Dynamic Web Applica-tion " , IEEE Int'l Conf. Software Eng., vol. 38, in 2012.
[2] S. Artzi, J. Dolby, F. Tip, and M. Pistoia, "Practical Fault Localization for Dynamic Web Applications," Proc. 32nd ACM IEEE Int'l Conf . Software Eng., vol. 1, pp. 265-274, 2010.
[3] Wenhong S. Artzi, A. Kie_zun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M.D. Ernst, "Finding Bugs in Dynamic Web Applications," Proc. Int'l Symp. Software Testing and Analysis, pp. 261-272, 2008.
[4] Artzi, A. Kiezun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M.D. Ernst, IBM Fault localization Re-search, in 2008.
[5] Artzi, J. Dolby, S. Jensen, A. Møller, and F. Tip, "A Framework for Automated Testing of JavaScript Web Applications," in Intl. Conference on Software Engineering (ICSE). ACM, 2011.
[6] K. Pattabiraman and B. Zorn, "DoDOM: Leveraging DOM Invariants for Web 2.0 Application Robustness Testing," in IEEE Intl. Symposium on Software Reliability Engineering (ISSRE). IEEE Computer Society,2010.
[7] Y. Zheng, T. Bao, and X. Zhang, "Statically locating web application bugs caused by asynchronous calls," in Intl. Conference on the World-Wide Web (WWW). ACM, 2011.
[8] Yong R. Abreu, P. Zoeteweij, and A.J.C. van Gemund, "An Evaluation of Similarity Coefficients for Software Fault Localization," Proc. 12th Pacific Rim Int'l Symp. Dependable Computing, pp. 39-46, 2006.
[9] S. Artzi, J. Dolby, F. Tip, and M. Pistoia, "Directed Test Generation for Effective Fault Localization," Proc. 19th Int'l Symp. SoftwareTesting and Analysis, pp. 49-60, 2010.
[10] Y. Zheng, T. Bao, and X. Zhang, "Statically locating web application bugs caused by asynchronous calls," in Intl. Conference on the World- Wide Web (WWW). ACM, 2011, pp. 805–814.
[11] M. Renieris and S. Reiss, "Fault localization with nearest neighbor queries," in Proceedings of the 18th International Conference on Automated Software Engineering (ASE). IEEE Computer Society , 2003
[12] F. Tip, "A Survey of Program Slicing Techniques," J. Programming Languages, vol. 3, no. 3 pp. 121-189, 1995
[13] J. Lyle and M. Weiser, "Automatic Bug Location by Program Slicing," Proc. Second Int'l Conf. Computers and Applications 2007.
[14] A . Mesbah and A. van Deursen, "Invariant-based automatic testing of AJAX user interfaces," in Intl. Conference on Software Engineering (ICSE). IEEE Computer Society , 2012
[15] S.Artzi ,J . Dolby, S. Jensen, A . Moller , and F . Tip , " A Framework for Automated Testing of JavaScript Web Applications ," in Intl. Conference on software Engineering (ICSE) .ACM 2011 , pp.-571 -580 .