Fault Detection Technique for Dynamic Web Applications

D.RajyaLakshmi[1], L.Abdul Saleem[2]

[1]M.Tech Student, Dept of CSE, Sridevi Engineering College, Gandipet, Hyderabad, A.P, India

[2]Assistant Professor, Dept of CSE, Sridevi Engineering College, Gandipet, Hyderabad, A.P, India

Abstract: We created associate degree algorithmic rule that generalizes hypertext markup language validation of individual documents to figure on context-free sets of documents in conjunction with a program analysis that soundly approximates the output of Java Servlets and JSP web applications as context-free languages, we tend to acquire a way for statically checking that such net applications sometimes turn out invalid hypertext markup language at runtime. An experiment with our model implementation demonstrates that the approach is useful and produces elaborate error messages that facilitate the applied scientist find the sources of the errors. once manually correcting the errors reportable by the tool, the soundness of the analysis ensures that no additional validity errors exist within the applications. Finally that process easily identifies the Html tag missing error and java script error or even java based logical or functional errors.

Keywords: HTML Validate, Bug Reporter, Junit tool, Lunuo tool.

I.INTRODUCTION

An important part of diagnosing and repair consists in localizing faults, and several other tools for machine-controlled de- bugging Associated in Nursing systems. Diagnosing implements Associate in nursing approach to fault localization supported by analysis of the differences in program spectra [20] for passed and unsuccessful runs. Passed runs square measured executions of a program that completed properly, whereas unsuccessful runs square measured executions within which a mistake was detected. A program spectrum is Associated in Nursing execution part that indicates that elements of a program square measure active throughout a run. The matter with current watching systems is that they start with the worst-case assumption that almost something may be a clue for a bug, so continue watching events even when applied math analyses show that almost all don't seem to be prognostic of failure. Distinguishing this with the targeted debugging activity Associated in nursing skilled engineer. Victimization feedback from a previous execution, or perhaps simply Associate in nursing initial hunch, the engineer uses breakpoints and different probes close to points of failure to urge additional feedback regarding program behavior. Suspect code is examined more closely, whereas orthogonal code is quickly known and unnoticed. This paper extends to take a look at generation of scripting languages, uses algorithms in detecting to work out whether or not the output of the net application is syntactically correct, and mechanically types and minimizes the inputs that expose errors which may be functional also. The output of an internet application is often associated in nursing markup language page which will be displayed in a browser. Our goal is to search out errors that crash internet applications, or ends up in an ill-shaped markup language. Some errors might terminate the appliance, like once an internet application calls Associate in nursing undefined function or reads a nonexistent file. In such cases, the markup language output presents a mistake message displayed within Associate in nursing obtrusive table and therefore the program execution is halted. Such applications generate structured output within the sort of dynamically generated markup language pages that will seek advice from further scripts to be read.

Like any program, programmers build mistakes and introduce faults, leading to Web-application crashes and ill-shaped dynamically generated markup language pages. While not able to get that info, correcting these kinds of problems is terribly troublesome.

This paper addresses the matter of deciding wherever within the ASCII text file changes have to be compiled or to be created so as to repair the detected failures. Before presenting our adaptive approach, we have a tendency to review basic ideas and methods employed in previous work on applied math debugging, together with the recent HOLMES [7] project that uses coarse-grain method adaptively. We have a tendency to use a activity model supported as that of the Cooperative Bug Isolation Project (CBI) of Liblit. We enforced three techniques in Lunuo Tool, an automatic tool for investigation of failures and localizing faults in JSP pages. This is associated by the Junit.

We have used Lunuo Tool to localize Dynamic faults indiscriminately selected faults in four JSP applications and compared the effectiveness of the three fault localization techniques. Our findings show that, victimization our greatest technique, 87.7% of the faults square measure localized to among a hundred and twenty fifth of all read statements that constitutes Associate in nursing virtually five-fold improvement over the Tarantula formula. We have a tendency to gift three fault localization techniques that mix variations on the Tarantula formula, Jaccard and Ochiai algorithms with the employment of Associate in nursing output mapping from statements to the fragments of program output that they turn out. victimization our greatest technique, that augments the domain of Tarantula for conditional statements and uses the output mapping to fine-tune Tarantula's distrustfulness ratings, the engineer must explore solely 2% of the read statements, on average.

II.RELATEDWORK

Automated Testing:

Current follow in takes a look at JavaScript internet applications and needs manual construction of test cases that is tough and tedious. We have a tendency to gift a framework for feedback-directed machine-controlled take a look at generation for JavaScript during which execution is monitored to gather data that directs to take a look at generator towards inputs that yield enlarged coverage, adore variations on feedback-directed random testing, in an exceedingly useful tool referred to as Greek deity. Experiments on a collection of JavaScript applications demonstrate that an easy representation of the framework that uses event handler registrations as feedback data produces amazingly smart coverage if enough tests are generated. By additional victimization of coverage data and read-write sets as feedback data, a rather higher level of coverage are often achieved, and generally with several fewer tests. The generated tests are often used for detection of HTML validity issues and alternative programming errors.

Algorithms like Ochiai,Jaccard and Tarantula have been investigated on dynamic JSP pages for functional errors with fault localization graphs.

Adaptive Bug Isolation:

Statistical debugging uses light-weight instrumentation and applied mathematics models to spot program behaviors that area unit powerfully prognosticative of failure. However, most packages is usually correct; nearly all monitored behaviors area unit poor predictors of failure. Performance measurements show that adaptational bug isolation yields a median performance overhead of one hundred and twenty fifth for a category of huge applications, as opposition eighty seven for realistic sampling-based instrumentation and three hundredth for complete binary instrumentation Throughout this method, statistical-analysis results area unit obtainable to the computer user, WHO will fix failures if enough knowledge is accessible or additional highly to|favor to|opt to} look forward to more knowledge if the image is unclear. Effectively, we have a tendency to replace sampled measuring of all predicates with non-sampled measuring of adaptively-selected predicates. Non-sampled instrumentation permits quicker adaptation by quickly gathering spare knowledge wherever it's most required. adaptational instrumentation improves upon existing approaches by prioritizing the observation of probably helpful behaviors over people who area unit less helpful, thereby preserving procedure resources and information measure for each users and developers.

Binary Instrumentation:

Each website} is rotten into atiny low assortment of instrumentation predicates that partition the state area at that site. At a branch instrumentation web site, we have a tendency to distinguish between executions that continue on truth versus the false branch. Thus, a branch web site decomposes into 2 predicates. A perform come instrumentation web site resides within the caller simply when the referred to as perform returns. Every perform come web site decomposes the state area into 3 subspaces, adore 3 predicates, counting on whether or not they came back price is negative, zero, or positive. This partition is very compatible to C programs, because the sign of a came back price typically indicates success or failure of associate operation Events. Instead, predicates area unit counted: one counter per predicate, incremented once that predicate is determined to be true. as a result of every web site's predicates partition the area of potentialities at that site, every observation of 1 instrumentation web site increments precisely one predicate counter. Thus, the add of all predicate counters at a web site provides the coverage of the positioning. Liblit et al. [26] argue that one mustn't even collect complete predicate counts. Instead, they provide a sampling theme supported
a static source-to-source transformation applied at compile time. Our adaptational approach eschews sampling in favor of complete measuring of a a lot of selective set of all attainable instrumentation. A feedback report within the sites-and-predicates model consists of a vector of all predicate counts and one outcome label marking this run pretty much as good (successful) or unhealthy (failed). Within the simplest case, failures are often outlined as blooming, and success as not blooming. a lot of refined labeling ways area unit simply accommodated, as succeeding analysis stages don't care however the success/failure distinction was created. Particularly, failure analysis doesn't use stack traces, and so are often applied to non-crashing bugs.

III.SYSTEM MODULES

W3c markup validation service:
The Markup Validator could be a free service by W3C that helps check the validity of internet documents. Most internet documents ar written exploitation markup languages, like hypertext markup language or XHTML. These languages ar outlined by technical specifications that typically embody a machine-readable formal synchronic linguistics (and vocabulary). The act of checking a document against these constraints is named validation, and this is often what the Markup Validator will. corroboratory internet documents is a very important step which may dramatically facilitate up and guaranteeing their quality, and it will save plenty of your time and cash (read a lot of on why corroboratory matters). Validation is, however, neither a full quality check, neither is it strictly reminiscent of checking for conformity to the specification. This validator will method documents written in most markup languages. Supported document varieties embody the hypertext markup language (through HTML four.01) and XHTML (1.0 and 1.1) family, Math ML, SMIL and SVG (1.0 and 1.1, as well as the mobile profiles). The Markup Validator may validate internet documents written with AN SGML or XML DTD, provided they use a correct document sort declaration.

To create the home page for Don's Cafe:

1. Create a new document in your text editor and type the opening <! DOCTYPE> declaration that uses the HTML DTD, as follows:
<! DOCTYPE HTML>
2. Type the <html> element, as follows.
<html> </html>
3. Within the <html > element, add the following <head> and <body> elements:
<head> <meta http-equiv="content-type" content="text/htmI; charset=utf-8" />
<title>Don's Cafe</title> <link href="style.css" rel="stylesheet" type="text/css" / </head>

```
<body>
</body>
```

4. Add the following elements to the document body. This Web page uses <div> tags with associated styles to lay out and format the document.

```
<div id="mainbody">
<div id="header">
<img src="images/header gif" width="761"
height="345" alt="" /></div>
<div id="contentarea">
<img src="images/about_service.gif" />
<p>&nbsp;</p>
<img src="images/special_service.gif" />
<p>&nbsp;</p> <br class="clearcols" />
</div> <div id="copyright">
<p>Copyrighted &copy; by 2011 Don's Cafe</p> <p
class="copy">All Rights Reserved</p>
<br class="clearcols" />
</div></div>
```

To add a script section to the Menu page for Don’s Cafe:

1. Return to the index.html document in your text editor.
2. Locate the first paragraph and nonbreaking space (&nbsp;) and replace it with the following script **section:**

```
<script type="text/javascript">
</script>
```

3. Save the index.html document

.

To add some variables to the script in index.html:

1. Come back to the index.html go in your text editor.

2. Add the subsequent volt-ampereiables to the top of the script section: var blue Mountain = "This extraordinary low, notable for its exquisite flavour and robust body, is adult within the majestic Blue chain in Jamaica."
var blueGrove = "This pleasant low has AN aroma that's enthrallingly made and nutty with a faint hint of citrus."
var island = "One of the best coffees within the world, medium roast to intensify its sturdy character."
volt-ampere kona = "Grown and processed victimization ancient Hawaiian strategies, then roast in tiny batches to keep up peak freshness and flavour."

var Antigua = "An outstanding low with an upscale spicy, and smokey flavour."

3. Save the index.html document. Later during this section, you may add code that displays the values allotted to every of the variables you more within the last step.

A. User Registration:

This module is that the user registration module, wherever users area unit registered with Product supplier (PP). The user must register with Product supplier to access the information. The merchandise supplier contains one web site to urge the complete user details for the privacy. The user income for registration through the registration web site managed by PP. The user enters his name, email address, contact range and different details, and additionally has to be compelled to settle for the terms and conditions that area unit given by PP.

B. Failure Detector:

We developed this module that may scan the input directory and checks for the JSP pages. When retrieving all the pages, the module analyzes the configuration file and understands the project flow. Then it starts from the beginning page and analyzes the code and splits into markup language and java contents and sends these to 2 completely different failure detectors. The markup language Validate is enforced by mistreatment Offline W3C Markup Validation Service. The markup language content also can be valid by mercantilism markup language validation packages that area unit provided by the Java Enterprise Edition API.

The Execution Failure Detector is enforced so as to see for the foremost common exceptions. FileNotFound Exception may be copied by checking out the file name that the jsp page is flowing into and compare with the list of jsp files. If the name isn't found the exception is raised. Similarly, when analyzing the net.xml, checking for category|the category} names and comparison of those class names with the list of sophistication files that area unit placed in /WEB-INF/classes folder may be done. By tracing the category objects, we've got analyzed all the potential ways that area unit gift in supply files and brought these names into repository and by perceptive the flow, we've got copied the MethodNotFound Exception.

C.HTML Validate:

The hypertext markup language Validate is chargeable for supportive the hypertext markup language content per W3C Markup Validation Service. Although we have a tendency to ar victimization offline W3C validation service, the final validations are often done by mercantilism and victimization the hypertext markup language packages.

Random informatics List:

We developed a program to every which way generate AN IPv4 address. By fixing AN hypertext transfer protocol association to the present informatics address at port eighty, that is that the default port for hypertext transfer protocol connections, and taking note of the response, we'd understand whether or not it had been an online server at this informatics address. If it had been so an online server, we have a tendency to more it to the list. we have a tendency to encountered some difficulties victimization this approach. Since most informatics addresses were addresses for on-line computers that weren't
Web servers, most of our probes would trip. once a brief whereas our service supplier received complaints concerning our activities since our probes were thought of hacker's probes by some organizations. We have a tendency to had to prevent the gathering method once getting an inventory of solely concerning one, 100 informatics addresses.

Most Popular ten thousand Websites List:

There are many URL lists accessible on the net, like the "500 hottest websites" from Alexa [2]. The validity of such websites is vital since, if hierarchic properly by the list providers, they're oftentimes accessed by people or computers. Their properties have higher impact on rendering, accessibility, and ability problems than a webpage designated randomly. We have a tendency to appreciate the support of Alexa.com who, once understanding the goals of our research provided North American nation with the foremost well-liked ten thousand Websites List. The results obtained victimization this list supported our findings victimization the Random informatics List.

URL List:

Both the Random information processing List and also the Most in style ten thousand Websites List allowed U.S. to ascertain the house pages of those websites solely. Since homepages represent solely a little portion of the complete net, we tend to want to review the properties of different sites to check if they're in keeping with the properties of homepages.

D. Bug Reporter:

Bug Report Repository This repository stores the bug reports found altogether executions. When a failure is detected, the corresponding bug report (for all failures with identical characteristics) is updated with the trail constraint and also the input causing the failure. A failure is outlined by its characteristics, that include: the sort of the failure (execution failure or hypertext markup language failure), the corresponding message (JSP error /warning message for execution failures, and validator message for hypertext markup language failures), and also the statement generating the problematic hypertext markup language fragments known by the validator (for hypertext markup language failures), or the JSP statement concerned within the JSP Compiler error report (for execution failures). Once the exploration is complete, every bug report contains one failure characteristics, (error message and statement concerned within the failure) and also the sets of path constraints and inputs exposing failures with identical characteristics.

We hope our findings square measure helpful to the quality bodies and authoring tool vendors thought to review the present practices and reach agreement on that options should become a part of the quality, and that options ought to not be supported by authoring tools. The authoring tools might do higher by together with the feature of some Tidy tools before the work is saved minimum of they ought to not give hints that may recommend the employment of invalid options whereas the page is being created. If all authoring tools will support that the validity check before saving a document, not solely new web content are going to be valid however conjointly some existing web content, once updated or traced victimization the authoring tool, are going to be valid conjointly. It's solely through the higher cooperation of normal bodies and authoring tool vendors will the net reach its full potential.

Experimental Results:

From the we tend to pages we tested; our program was able to catch concerning ninety eight of the whole errors according by the W3C Validator. Figure a pair of shows the error distribution within the Random informatics List. Note that our list includes the six "common" issues [15] known by WDG. A additional elaborated analysis shows that variety of} these six issues aren't moving a major number of existing webpages. We tend to make a case for a number of the foremost common errors below. All the figures shown during this section square measure supported the knowledge from the Random informatics List. There square measure many attributes needed in specific elements: for example, the attribute "alt" of "IMG" and also the attribute "type" of "SCRIPT". Since omitting these attributes causes no rendering difficulties for browsers, webpage designers sometimes neglect to embrace them. In Figure three, we tend to found that almost four-hundredth of web content omitted the attribute "alt" whereas half-hour of web content omitted "type". Even once it provides the "alt", the worth of "alt" is usually associate in nursing empty string.
We conjointly noticed that almost hour of the component "SCRIPT" omitted "type". These 2 sorts of attribute errors created up ninety nine of the Missing needed Attributes errors. concerning hour of all the net pages have this type of errors. A take a look at on the standard of the text for the "alt" attribute is providedhttp://www.hisoftware.com/accmonitorsitetest/. By victimization this validate, we discover that solely thirty eighth web content within the Random informatics List passed this take a look at, terrorist organization web content passed the take a look at with some warnings, and also the remaining four hundred and forty yards failing the take a look at. The hypertext mark-up language customary needs quotation marks around literal values of attributes. for instance, a white background color ought to be mere as "bgcolor="#FFFFFF"". though most browsers will render the color properly with or while not quotation marks, they will render otherwise in alternative things such as "size="+1"" and "size=+1". so missing quotation marks might cause a webpage to become invalid. we tend to were stunned to notice that once Microsoft net mortal is employed to save Associate in Nursing hypertext mark-up language document victimization the perform "Save As web content, complete", it removes most of the quotation marks around literal values

Java functional errors in the failure detector are analyzed and the following results are obtained after running dynamic JSP web pages using Lunuo Tool in J2EE eclipse environment. The JUnit assists Lunuo tool in finding Logical errors in the Dynamic JSP content of the web application.

Test Results:

Rank	Suspicious	Line	Class
2	1.0[1.0]	9	FindMid
2	1.0[1.0]	8	FindMid
3	8.33[1.0]	7	FindMid
4	7.14[1.0]	5	FindMid
8	0.5[1.0]	14	FindMid
8	0.5[1.0]	4	FindMid
8	0.5[0.1]	3	FindMid

8	0.5[1.0]	1	FindMid
9	0.0[0.6]	10	FindMid
10	0.0[0.4]	12	FindMid
13	0.0[0.2]	13	FindMid
13	0.0[0.2]	11	FindMid
13	0.0[0.2]	6	FindMid

Fig.1 Tarantula Results

Rank	Suspicious	Line	Class
2	1.0	9	FindMid
2	1.0	8	FindMid
3	0.5	7	FindMid
4	0.3333334	5	FindMid
8	0.166667	14	FindMid
8	0.166667	4	FindMid
8	0.166667	3	FindMid
8	0.166667	1	FindMid
13	0.0	13	FindMid
13	0.0	12	FindMid
13	0.0	11	FindMid
13	0.0	10	FindMid
13	0.0	6	FindMid

Fig.2 Jacard Results

Rank	Suspicious	Line	Class
2	1.0	9	FindMid
2	1.0	8	FindMid
3	0.7071067	7	FindMid
4	0.5773502	5	FindMid
8	0.4082483	14	FindMid
8	0.4082483	4	FindMid
8	0.4082483	3	FindMid
8	0.4082483	1	FindMid
13	0.0	13	FindMid
13	0.0	12	FindMid
13	0.0	11	FindMid
13	0.0	10	FindMid
13	0.0	6	FindMid

Fig.3 Ochiai Results

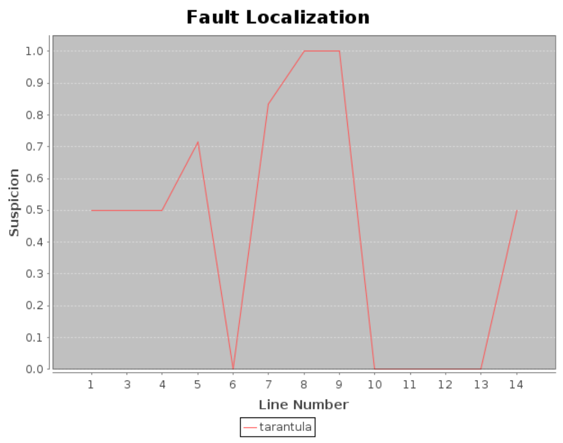

Fig4. Tarantula Results Graph

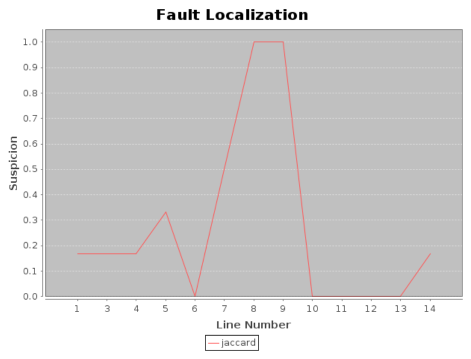

Fig5. Jaccard Results Graph

IV.CONCLUSION

The validation issues in existing webpages have drawn a lot of attention. Thanks to the increasing trend of internet communications moving from a human-to-computer method to a computer-to-computer method. This paper attempts to conduct associate experiments that are known as the main validation issues in current webpages. We tend to use the Random information processing List as a basic check sample and establish the highest ten issues that cause the webpages to fail the validator. The Dynamic content of sample JSP pages has been tested using various algorithms of the Lunuo Tool , namely Ochiai, Jaccard and Tarantula are used as a comparison for better automated testing and the results have been analyzed and found that depending on the application, the Jaccard and Ochiai algorithms are found to be more efficient in many cases rather than the tarantula algorithm.As a future enhancement, we can apply to any HTML,and jsp/PHP or .NET applications.Finally Bug report generated this one easily identify Html tag missing function,and java script link error process.

ACKNOWLEDGMENT

We wish to acknowledge the efforts of **Pantech Solution Pvt ltd., Hyderabad,** for guidance which helped us work hard towards producing this research work.

V.REFERENCES

[1] S. Artzi, J. Dolby, F. Tip, and M. Pistoia. Practical faultlocalization for dynamic web applications. In *Proc. 32nd Int.Conf. on Software Engineering, ICSE '10*, May 2010.

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

[3] P. Boonstoppel, C. Cadar, and D. R. Engler. RWset:Attacking path explosion in constraint-based test generation.In *Proc. 14th Int. Conf. on Tools and Algorithms for theConstruction and Analysis of Systems, TACAS '08*,March-April 2008.

[4] R. C. Bryce and A. M. Memon. Test suite prioritization byinteraction coverage. In *Proc. Workshop on Domain Specific Approaches to Software Test Automation, DOSTA '07*,
September 2007.

[5] C. Duda, G. Frey, D. Kossmann, R. Matter, and C. Zhou.AJAX crawl: Making AJAX applications searchable. In*Proc. 25th Int. Conf. on Data Engineering, ICDE '09*,March-April 2009.

[6] ECMA. ECMAScript Language Specification, 3rd edition. ECMA-262.
[7] P. Godefroid, N. Klarlund, and K. Sen. DART: Directedautomated random testing. In *Proc. ACM SIGPLAN Conf. onProgramming Language Design and Implementation,PLDI '05*, June 2005.

[8] S. Guarnieri and B. Livshits. Gatekeeper: Mostly staticenforcement of security and reliability policies for JavaScriptcode. In *Proc. 18th USENIX Security Symposium*, August
2009.

[9] A. Guha, S. Krishnamurthi, and T. Jim. Using static analysisfor Ajax intrusion detection. In *Proc. 18th Int. Conf. onWorld Wide Web, WWW '09*, April 2009.

[10] P. Heidegger and P. Thiemann. JSConTest: Contract-driventesting of JavaScript code. In *Proc. 48th Int. Conf. onObjects, Components, Models and Patterns, TOOLS '10*,LNCS. Springer, June-July 2010.

[11] H.-Y. Hsu and A. Orso. MINTS: A general framework andtool for supporting test-suite minimization. In *Proc. 31st Int.Conf. on Software Engineering, ICSE '09*, May 2009.

[12] S. H. Jensen, A. Møller, and P. Thiemann. Type analysis forJavaScript. In *Proc. 16th Int. Static Analysis Symposium,SAS '09*, volume 5673 of *LNCS*. Springer, August 2009.

[13] A. Le Hors et al. Document Object Model (DOM) level 3core specification, April 2004. W3CRecommendation.http://www.w3.org/TR/DOM-Level-3-Core/.

[14] A. Marchetto and P. Tonella. Search-based testing of Ajaxweb applications. In *Proc. 1st Int. Symp. On Search BasedSoftware Engineering, SSBSE '09*, May 2009.

[15] A. Marchetto, P. Tonella, and F. Ricca. State-based testing ofAjax web applications. In *Proc. 1st Int. Conf. on Software Testing, Verification, and Validation, ICST '08*, April 2008.

[16] A. M. Memon. An event-flow model of GUI-based applications for testing. *Software Testing, Verification & Reliability*, 17(3):137–157, 2007.

[17] A. Mesbah, E. Bozdag, and A. van Deursen. Crawling AJAXby inferring user interface state changes. In *Proc. 8th Int. Conf. on Web Engineering, ICWE '08*, July 2008.

[18] A. Mesbah and A. van Deursen. Invariant-based automatictesting of AJAX user interfaces. In *Proc. 31st Int. Conf. onSoftware Engineering, ICSE '09*, May 2009.

[19] C. Pacheco, S. K. Lahiri, M. D. Ernst, and T. Ball.Feedback-directed random test generation. In *Proc. 29th Int.Conf. on Software Engineering, ICSE '07*, May 2007.

[20] G. Richards, S. Lebresne, B. Burg, and J. Vitek. An analysisof the dynamic behavior of JavaScript programs. In *Proc.ACM SIGPLAN Conf. on Programming Language Design and Implementation, PLDI '10*, June 2010.

[21] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold.Prioritizing test cases for regression testing. *IEEE Trans. onSoftware Engineering*, 27(10):929–948, 2001.

[22] P. Saxena, D. Akhawe, S. Hanna, S. McCamant, D. Song,and F. Mao. A symbolic execution framework for JavaScript. In *Proc. 31st IEEE Symp. on Security and Privacy, S&P '10*,May 2010.