International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 5, May - 2014

Automation in Testing of Modern Web Applications

Mr. Satyajit S. Nimbalkar
Research Scholar, Department of Computer Engineering,
Vidya Pratishthan College of Engineering, Baramati, 413 133,
Dist-Pune, University of Pune, Maharashtra, India

Prof. Santosh S. Shinde
Assistant Professor, Department of Computer Engineering,
Vidya Pratishthan College of Engineering, Baramati, 413 133,
Dist-Pune, University of Pune, Maharashtra, India

***Abstract*-The Modern computer's entire world has a lot's of Internet apps. In that blogging platform will be the most famous essential technologies. Web 2.0 contains AJAX dependent app that's stateful asynchronous client-server connection. It truly is harder to evaluate with regard to tester this sort of internet apps due to its invariant dynamics of these apps. Safety measures is additionally the main matter due to presence of broken enter through the tester. And so we recommend a method with regard to assessment of AJAX Internet app with supplying stability towards created analyze circumstances. Planned method is dependent on Internet crawler. Each of our suggested system's job will be to uncover wrong doing linked AJAX apps, including mistake mail messages, rear option compatibility, DOM tree agreement, in addition to broken enter with stability. Planned method implementation contains state age group; analyze suite age group with covering the overall journey produce with the crawler, age group of stability exams by making use of proper danger versions. Safety measures procedure utilizes quantity of stability mutants wherever vulnerabilities are usually being injected intentionally. These mutants were created keeping that in mind their vulnerabilities. This specific app utilizes wherever require of automation which in turn having fantastic wrong doing locating, less manual energy in addition to scalability.**

Keywords—Web Application, Ajax, Automation in Testing, Security Testing, Threat Modeling, Model Based Testing, mutation Testing, Petri nets

Introduction

AJAX (Asynchronous JAVASCRIPT and XML) requisitions which are dynamic in nature give rich and quick communication with UI of HTML for the clients. Some security ambushes additionally come about because of client's unattended practices or invalid info. A few tool like Selenium, Sahi offers AJAX base testing yet these tools still oblige manual exertions with less security [1]. Consequently, our objective to create robotized testing tool with testing systems and likewise distinguishing programming vulnerabilities Ajax requisitions oblige dynamic examination for helpful for such AJAX web testing. CRAWLJAX is open source tool to crawling the web requisitions. This module gives a model having client interface state of web requisitions. CRAWLJAX module utilized for discovers the AJAX states as far as DOM tree and substantial HTML records [1]. In the wake of creating of HTML archive the experiment are designed. This methodology is helpful for flaw discovering competence of web provision and also server. Risk model spoke to as Predicate and Transition nets [3]. Secure model creates the assault way. Secure experiments are produced utilizing test code applying change based testing. The center of security framework is on security test code. Risk model determination has methodology of execution by utilizing of Integration and framework Test Automation.

I. LITERATURE SURVEY

Ajax web application is challenging in nature due to its dynamic behavior. Following some tools like Veriweb, Waves, SecuBat, Mcweb, Selenium are used for web application testing.

A. Review of Web Based Testing

WAVES is trying stages which give security testing [9]. WAVES having ability of figuring out security ambush. It recognizes information section focuses. It likewise gives infusing noxious examples screen to programs. WAVES having impediments about invariant proficiency and crawler does not have record dynamic conduct. MCWEB model checking tool utilized for web 1.0 provision [11]. These tools fundamentally concentrate on site at outline level. It checks neighborhood and worldwide level outline of site. MCWEB check configuration like edge. MCWEB having not having given data information level practicality. MCWEB not utilized for element sites.

Veriweb testing apparatus for the most part concentrated on adaptable route purpose with catch replay tool [8]. Veriweb obliged quest calculation for sites. Veriweb tool has issue of element nature site.

Reweb apparatus utilized for making model of web provision in UML. Reweb require to characterized scope criteria for keep up experiments. A tool developed by Andrew which is focused around test model. Test criteria information to produce experiment. All above tools utilized more established courses for web creeping. Apollo Arch which utilized testing of PHP provision focused around joined solid typical execution. This tool is great in flaw discovering capacity of PHP requisition, such kind of hardware having impediments against the complex element site.

B. Current technology for AJAX Testing

AJAX testing requisition could be tried by some customary testing tool strategy. AJAX testing might be performed on distinctive levels. In the present web testing innovation basically late tools utilized are for trial, for example, Selenium IDE, Sahi, and Web King. These tools are takes after DOM based testing. These tools are catching occasions which are terminated by client at client interface level.

Selenium offers cutting edge web provision testing. Selenium apparatus gives catch and record all data of web [15]. At long last prepare yield as experiments. Still selenium obliges manual exertions for analyzer. Our framework methodology is to uproot these manual endeavors hindrances.

II. PROBLEM STATEMENT AND IMPLEMETATION DETAILS

A. Problem Statement

To design an Automatically Testing for modern AJAX web application, which is based on open-source crawler CRAWLJAX and provides plug-in hooks for testing AJAX applications at different levels enhance the current testing tool security by using formal threat model.

B. Problem Solving Approach

Invariant is an idea which utilized to focus conduct of AJAX system. Invariant intends to attest system conduct of AJAX when project is in running state. This idea is utilization investigation for element conduct. AJAX testing having tests to discover dynamic client collaboration and element DOM states. The occasion driven is nature of AJAX provisions are tests to clients. AJAX requisitions reaction and solicitation will be mimics. Web Crawler gives result on the recognizing and terminating occasions by recording clickable occasions. Web crawler is utilized for slithering the sites. Web Crawler ought to fit to catch the clickable occasions which haphazardly gave by the client. Route way and UI satiates these two parts give paramount errand to the crawler [1]. CRAWLJAX assume a critical part in outlining AJAX testing tool. CRAWLJAX holds two fundamental calculations for Pre slithering and Post Crawling. The slithering procedure gives the state stream chart holds points of interest data about the UI states.

C. System Modeules

The Project Design contains two main parts

1) Design of Automatic Testing Tool

2) Threat Based Model for Test Case Security

1) Automatic Testing Tool Contains following modules

a) Generating State flow Graph
b) Inferring the State Machine
c) Find out click able events
d) Creation and Comparison between States
e) Process Document DOM tree Deltas
f) Search for Navigations the States
g) Input Data entry point
h) Control over the Crawling Phase
i) Testing AJAX Test through Invariant
j) Test Suite Generation
k) Test-Case Execution

2) Threat Based Model Design contains following modules

a) Design of Threat Model
b) Model- Implementation Mapping
c) Generating Attack Paths with Security Test Cases.

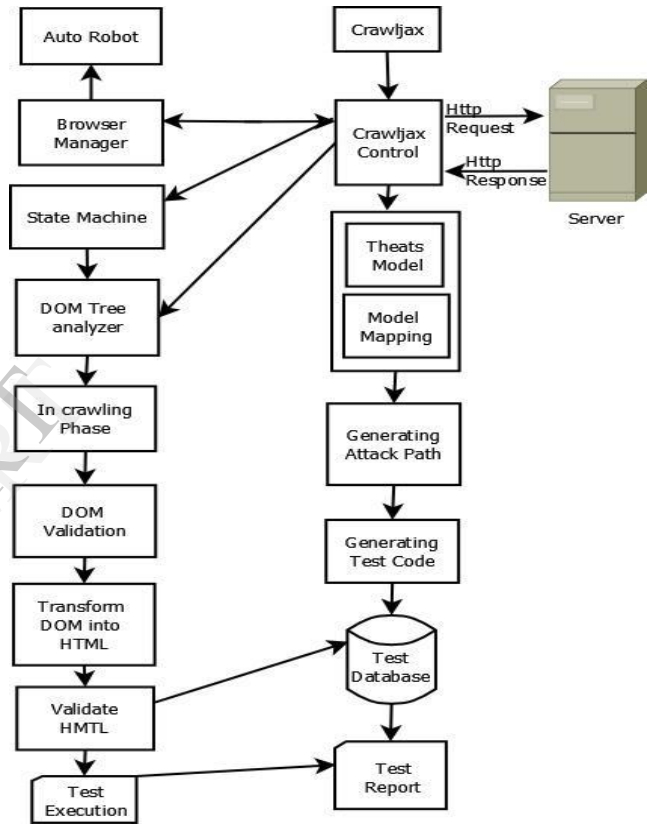

Fig.1 System Architecture

1) Generating State flow Graph

Crawljax is tool which can make outline of state of customer side code and distinguish component. The progressions are made which will be discovering progressively from DOM tree components. State stream diagram is only occasion base moves [2]. State stream chart is characterized AS Ajax Site with three tuple (r, V, E)

• V is situated of state vertex which is recognize client interface component of AS.

• r is root component hub which is begin of list when

As is burden into program.

- e is the edges associated with state vertex after click capable events.

Consider below example for state flow graph AS the state flow graph for AJAX website in which index page with directly reachable three states [2].

Algorithm 1.Crawling process with pre/post crawling hooks [1]

1: procedure START (url, Set tags)

2: browser → initEmbeddedBrowser(url)

3: robot → initRobot()

4: sm → initStateMachine()

5: preCrawlingPlugins(browser)

6: crawl(null)

7: postCrawlingPlugins(sm)

8: end procedure

9: procedureCRAWL(State ps)

10: cs → sm.getCurrentState()

11:Δ update → diff (ps, cs)

12: f → analyseForms(Δupdate)

13: Set C → getCandidateClickables Δ update, tags, f)

14: for c ∈ C do

15: generateEvent (cs, c)

16: end for

17: end procedure

Algorithm 2.Firing events and analyzing AJAX states [1]

1: procedureGENERATEEVENT(State cs, Clickable c)

2: robot.enterFormValues(c)

3: robot.fireEvent(c)

4: dom → browser.getDom()

5: if stateChanged(cs.getDom(),dom) then

6: xe → getXpathExpr(c)

7: ns → sm.addState(dom)

8: sm.addEdge(cs, ns, Event(c, xe))

9: sm.changeToState(ns)

10: runOnNewStatePlugins(ns)

11: testInvariants(ns)

12: ifstateAllowedToBeCrawled(ns) then

13: crawl(cs)

14: end if

15: sm.changeToState(cs)

16: ifbrowser.history.canGoBackthen

17: browser.history.goBack()

18: else

19: {We have to back-track by going to the initial state.}.

20: browser.reload()

21: ListE → sm.getPathTo(cs)

22: for e ∈ E do

23: re → resolveElement(e)

24: robot.enterFormValues(re)

25: robot.fireEvent(re)

26: end for

27: end if

28: end if

29: end procedure

2) Interfering the State Machine

2)

CRAWLJAX contains pre and post crawling algorithm. At start of algorithm 1 input is **S**TART (url, Set tags) When crawler gets started different states are created. Basically it uses browser's driver. When system has input as a url browser should initialize

Browser → InitEmbeddedBrowser (url)

The state machine is created initially

sm → initStateMachine()

Input as a url browser should initialize. Browser initEmbeddedBrowser(url)

The state machine is created initially sm → initStateMachine()

At start of state machine it contain root element when new state is created element is added. Automatic Robot from CRAWLJAX is used which check click and text input from the

browser controller. From the browser the system have got DOM tree and states Analysis and checking for whether there is change in state. Browser also control over the different action taken by the Robot.

3) Find out clickable events

CRAWLJAX can check for the clickable events like e.g. OnClick in JavaScript. In the system to check out the clickable event at runtime is also critical task. CRAWLJAX uses candidate element for that. Crawl (Candidate Element) From Event detection algorithm, these candidate elements have labels like HTML tags. When user clickable elements are find out edges are created and added to state machine.

This will connect to previous state by using

IfstateChange (cs:getDOM; DOM)

Sm: changeTostate(NextState)

4) Creation and Comparison between States

During creation of states if same state which is already have state machine should be identify. If next state of graph is not in state machine graph of new state is created [6].

NextState → sm:Addstate(DOM)

To check out different states, it is possible use of comparator with the DOM tree. This DOM tree simply converted into strings and from strings it is easy to determine the equal state. String of state contains before list of comparator, attribute oracle comparator, Data oracle comparator and finally staring comparator

5) Process Document DOM tree Deltas

When new state is detected the crawler is recursively called method Crawl (Cs).To check out difference between candidate state: Δ Update State- Diff (Ps,Cs) Due to new request from server new element is produced so it is necessary to scan out such element.

6) Search for Navigations the States

After complication of self call the browser going into previous state. This previous state information is not given by back button from history stored information

xe → getXpathExpr(c)

ns → sm:addState(dom)

Xpath is used for to provide functionality to identify clickable event by using CRAWLJAX.re resolveElement(e) This statement used to detect and resolve the possible element in Xpath. Examination of click able events is done for to access correct element. Resolver search re resolveElement(e) This statement used to detect and resolve the possible element in Xpath. Examination of click able events is done for to access correct element. Resolver can search for DOM for a match.

7) Input Data entry point

User interface having input text value which provides input to AJAX application. CRAWLJAX detect the input type element from the form.f:Δ analyseForms(update)While checking form base value robot goes into input value. These input value having different type of categories

1) Random Input Value

2) Custom input Values

3) Multiple Customs input value

8) Control over the Crawling Phase

When crawl condition satisfied then crawl start for crawling. Condition is if stateAllowedToBeCrawled(ns) then When Crawling process get stated we must check conditions. Conditions are like check that state is visited or not. In the crawler controller you can provide maximum crawling time, waiting time and crawl depth.

9) Testing AJAX Test through Invariant

Web 2.0 contains dynamic web states so that some constraint applied on DOM element. Invariant is divided into two categories one is generic other is application based. a) Generic DOM with invariants

I) Check for Valid DOM

HTML code contains various malformed causes by browsers vulnerability. To validate these type of DOM is obtained from each state changes. Then DOM transform into corresponding HTML.

ii) No error message in DOM

A client side web doesn't contain error message like 404 pages not found. This type of fault can configure by tester [15].

iii) Secure states

Testing modern web applications for security vulnerabilities is far from trivial. Capturing web security requirements in terms of generic invariants that can be checked automatically is very promising. While perform that operation automatically detecting security vulnerabilities in client-side DOM for a match. Vulnerabilities are far from trivial. Capturing web security requirements in terms of generic invariants that can be checked automatically is very promising.

10) Test Suite Generation

To generate a test suite, use of the K shortest paths algorithm, which is like a generalization of the shortest path problem in which several paths in increasing order of length are sought. We get based collect all sinks in our graph, and to find the shortest path from the index page to each of them. Loops also are included once. This way, we can easily find all transitions coverage path. Given a rooted directed graph G with non negative edge weights, a positive integer K, and two vertices v1 and v2, the problem asks for the K shortest paths from v1 to v2, in non decreasing order of length. In our algorithm, first the set of sink vertices (with no outgoing edges) in G is calculated Then, our application use each sink in s1,s2,sn to find the K shortest paths from the root (index) state to si. Loops are included once. Next, our application transforms each path found into a JUnit test case [14].

Each test case captures the sequence of events from the initial state to the target state. The JUnit test case can fire events since each edge on the state-flow graph contains information about the event-type and the element the event is fired on to arrive at the target state. We also provide all the information about the clickable element, such as tag name and attributes, as code comments in the generated test method. The test class provides APIs to access the DOM (browser.getDom())

10) Test-Case Execution

Usually, extra coding is necessary for simulating the environment where the tests will be run, which contributes to the high cost of testing .Application can provide a framework to run all the generated tests automatically using a real web browser and generating success/failure reports. At the beginning of each test case, the embedded browser is initialized with the URL of the AJAX site under test. For each test case, the browser is first put in its initial index state. From there, events are fired on the clickable elements (and forms filled if present). After each event invocation, assertions are

checked to see if the expected results are seen on the web application's new UI state.

2) Design of Threat Model

This module of TMID gives, the front-end in-put language for automated security testing [3]. A TMID includes a threat model and a MIM specification. A threat model shows how at-tacks can be performed against the SUT, whereas a MIM gives specification maps the elements of a threat model to implementation-level constructs. The former is used to generate security tests and the used to convert them into executable code.

a) Threat Models Definition 1 (PrT net).A PrT net N is a tuple <P,T,F,I,∑,L,ℓ,M0 > ,where

P is a set of places (i.e., predicates), T is a set of transitions is a set of normal arcs, and I is a set of inhibitor arcs [5].

∑ is a set of constants, relations (e.g., equal to and greater than), and arithmetic operations (e.g., addition and subtraction).

L is a labeling function on arcs F U I. L (f) is a label for arc f. Each label is a tuple of variables and/or constants in ∑.

ℓ is a guard function on T:ℓ(t) t's guard condition, is built from variables and the constants, relations, and arithmetic operations in ∑.This formalism has been applied successfully to threat modeling in a formal method for secure software design

A PrT net <P, T, F, I, ∑, L, ℓ, M0 > is a threat model or net if T has one or more attack transitions (suppose the name of each attack transition starts with attack). The firing of an attack transition is a security attack or a significant sign of security vulnerability.

b) Model-Implementation Mapping

3.1) Let £ be the target language of test code (HTML/Selenium or C), $O^{£}$ be a set of expressions in £, $P^{£}$ be a set of code blocks in £. (MIM Specification).A MIM specification for a threat model N=<P,T,F,I,∑,L,ℓ,M0 > is a quadruple < SID, f_0 , f_{PT} , f_H >

SID is the identity or URL of the SUT.

f_0: ∑→ $O^{£}$ maps constants in ∑ to expressions in £.

$f_{PT:P\ U\ T\rightarrow}$ $P^{£}$:maps each place and transition in P U T to a block of code.

$f_{H=\{HEADER\}\rightarrow}$ $P^{£}$: is the header code in £ It will be included in the beginning of a test suite.

f_0 called object function, maps each constant (object or value) in a token, arc label, or transition firing of the threat net to an expression in the implementation [4].

3.2) Generating Attack Paths with Security Test

In a threat net, each attack path M_0,t_1 Mn (tn is an attack transition) is a security test, where

M_0 is the initial test setting;

t1 are test inputs.

M1...Mn-1 are the expected states (test oracles)

Algorithm 3.Generate security tests from a threat net [3].

Input: Threat net <P, T, F, I, ∑, L, ℓ, M0 >

Output: attackPaths - the list of all attack paths

1) Declare:root, newNode, currentNode are nodes;

2) Queue is a queue of nodes;

3) Substitutions (t, currentNode)are all substitutions that make t firable under currentNode.marking;

4) leafNodes is a list of leaf nodes;

5) attackPath(leaf)is the attack path from the root to the leaf;

needToRepeatLeafNodeExpansion is a Boolean variable, Initialized to true;

III RESULT

1) Data Set

Our Automatic Testing requires input Dataset as AJAX based web sites. These website are may be type of Application invariant based, Subject system base. In below example the application contain jquery, PHP based application.

2) Result Set

In result section we discus with Selenium and out Automatic testing Tool test cases [13].Selenium require manual effort while our tool remove this disadvantage.

a) Result using CRAWLJAX

{"id":"test1","name":"Test1","url":"http://www.google.co.in","browser":"FIREFOX","numBrowsers":1,"bootBrowser":true,"reloadWaitTime":500,"eventWaitTime":500,"maxDepth":2,"maxState":500,"maxDuration":1,"clickOnce":true,"randomFormInput":true,"clickDefault":true,"clickRules":[],"pageConditions":[],"invariants":[{"condition":"javascript","expression":"js"}],"comparators":[],"formInputValues":[],"lastCrawl":1399966967219,"lastDuration":111849,"lastModified":1399967079068,"plugins":[]}

Tests run: 1, Failures: 0, Errors: 0,
Skipped: 0, Time elapsed: 0.108 sec in
com.crawljax.core.state.ElementTest
Tests run: 2, Failures: 0, Errors: 0,
Skipped: 0, Time elapsed: 0.352 sec
in com.crawljax.test.WebServerTest
Tests run: 4, Failures: 0, Errors: 0
Skipped: 0, Time elapsed: 78.654 sec
in com.crawljax_plugins_plugin.CrawlersTest

b) Result using Security Threat Model

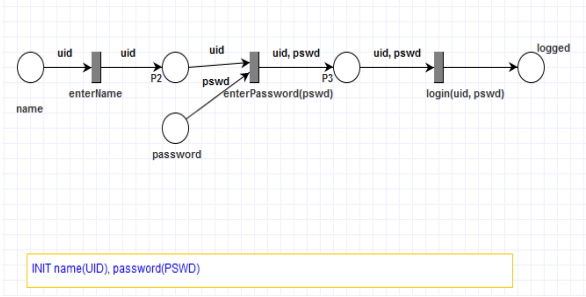

Fig.2 PrT net Model

Initial state: password(PSWD), name(UID)

1. enterName(UID)
   password(PSWD), P2(UID)
2. enterPassword(PSWD)
   P3(UID, PSWD)

Test code file: F:\Test\examples\robot\loginTester_RT.cpp
Search strategy: Breadth first; Maximum search depth: 20
Number of tests: 1; Number of states: 4; Depth of the deepest test: 3

1. t11, t12,t13(INJECTION1),attack
2. t11, t12,t13(INJECTION1),attack
3. t11, t12,t13(INJECTION1),attack

IV. Conclusion And Future Work

This testing application contains a method for testing AJAX applications automatically. Our starting point for supporting AJAX-testing is CRAWLJAX, which a crawler for AJAX applications. Our current work will consists of extending the crawler substantially for supporting automated testing of modern web applications. This approach have to automated security testing by using formal threat models This system uses novel algorithm for crawling process named as CRAWLAJAX which is the best approach to design our ATUSA tool. This system may for useful for secure software development projects with frequent changes of requirements Our Future work will include some multiple web application testing support, Web UI in for Crawling Testing. Our future work for Threat Based Model will apply more powerful analyzers/scanners to the security mutants. This may also helpful to determine how dynamic testing for security and static analysis for security.

Acknowledgment

The author acknowledges and thanks all the individuals who played defining role in shaping this journal paper. The authors must thankful open CRAWLJAX researcher to provide such crawling functions. Without their constant support, guidance and assistance this paper would not have been completed. Without their Coordination, guidance and reviewing this task could not be completed alone. Finally whole hearted thanks to author's guide Prof.S.A.Shinde for giving his valuable guidance, inspiration and encouragement to embark this paper.

.

References

[1] Ali Mesbah, Arie van Deursen, Danny Roest,"Invariant-Based Automatic Testing of Modern Web Applications," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 38, NO. 1. 35-53, 2012.
[2] A. Mesbah, E. Bozdag, and A. van Deursen "Crawling Ajax by Inferring User Interface State Changes", Proc. Eighth Intl Conf. Web Eng.,pp. 122-134, 2008.
[3] Dianxiang Xu,Manghui Tu,Michael Sanford,Lijo Thomas, "Automated Security Test Generation with Formal Threat Mod-els",IEEE TRANSACTIONS ON DEPENDABLE AND SE-CURE COMPUTING, VOL. 9, NO. 4,2012.
[4] Dianxiang Xu,"A Tool for Automated Test Code Generation from High level Petri Nets",NIC for Protection of Financial Infrastructure,2012.
[5] H.J. Genrich, "Predicate/Transition Nets Petri Nets: Central",Models and Their Properties,pp. 207-247, Springer-Verlag, 1987.
[6] 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.
[7] C.P. Bezemer, A. Mesbah, and A. van Deursen, "Automated Security Testing of Web Widget Interactions" ,Proc. Seventh Joint Meeting of the European Software Eng. Conf. and the ACM SIGSOFT Symp. the Foundations of Software Eng.,pp. 81-91, 2009.
[8] M. Benedikt, J. Freire, and P. Godefroid,"VeriWeb: Automatically Testing Dynamic Web Sites", Proc. 11th Intl Conf. World Wide Web, pp. 654-668, 2002.
[9] 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.
[10] S. Kals, E. Kirda, C. Kruegel, and N. Jovanovic," Secubat: A Web Vulnerability Scanner, Proc. 15th Intl Conf. World Wide Web,pp. 247-256, 2006.
[11] F. Ricca and P. Tonella,"Analysis and Testing of Web Applications", Proc. 23rd Intl Conf. Software Eng.,pp. 25-34, 2001.
[12] A. Mesbah, E. Bozdag, and A. van Deursen, "Crawling Ajax by Inferring User Interface State Changes , Proc. Eighth Intl Conf.Web Eng.,pp. 122-134, 2008.
[13] http://selenium.openqa.org
[14] http://jsunit.net.
[15] http://www.w3.org/TR/WAI-WEBCONTENT/