# Automatic Testing of AJAX Applications through Dynamic Analysis of User Interface State Change

Swapnil S. Mane
Research Scholar
Annasaheb Dange College
of Engineering & Technology,
Ashta, Tal-Walwa Dist-Sangli.

Amol B. Rajmane
Assistant Professor
Ashokrao Mane Group of Institutions,
Vathar tarf Vadgaon, Tal-Hatkanangale,
Dist- Kolhapur.

## ABSTRACT

The growing popularity and importance of web applications have been increasing continuously in recent years. Use of JAVASCRIPT and dynamic DOM (Document Object Model) manipulation on the client side of web applications is becoming a widespread approach for achieving rich interactivity and responsiveness in modern web applications. AJAX (Asynchronous JAVASCRIPT and XML) based web applications rely on asynchronous client-server communication and client-side runtime manipulation of the DOM tree. This not only makes them fundamentally different from traditional web applications but also make them more error prone and harder to test. The proposed method for testing automatically AJAX application is based on a crawler to infer a state-flow graph for all client-side user interface states of an AJAX application. Focus is on obtaining a model by "crawling" an AJAX application, automatically clicking buttons and other user interface elements. In order to recognize failures in executions, use of invariants are proposed [1]. These invariants can be generic (e.g., after any client-side change the DOM should remain W3C-compliant valid HTML) or application-specific (e.g., the home-button in any state should lead back to the starting state).

## General Terms

AJAX, JAVASCRIPT, XML, DOM

## Keywords

Crawler, event, traditional, modern, testing

## 1. INTRODUCTION

For today's web applications, one of the key facilitating technologies includes AJAX. With AJAX the web browsers not only offer the user navigation through a sequence of HTML pages, but also provide dynamic rich interaction via graphical user interface (GUI) components. Over the last three decades, the internet has become an essential part of everyday life. Users rely on the internet for tasks related to communication, information and commerce. In addition, the popularity of web-based applications is increasing with hundred millions of people. There is a need to crawl web applications (automatically discover all states of applications) and process them in various ways.

The web is undergoing a significant change. A technology that has lately gained a prominent position under the AJAX (Asynchronous JAVASCRIPT and XML) in which the combination of JAVASCRIPT and Document Object Model (DOM) manipulation along with asynchronous server communication is used to achieve a high level of user interactivity. Highly visible examples include Gmail and Google Docs. Following are the new challenges.

- Searchability -Searchability ensures that AJAX sites are crawled and indexed by the general search engines.
- Testability-Testability involves systematically dynamic user interface (UI) elements

One way to address these challenges is through the use of a crawler that can automatically crawl different states of a highly dynamic AJAX site and create a model of the navigational paths and states.

General web search engines, such as Google and Bing, cover only a portion of the web called the publicly indexable web that consists of the set of web pages reachable using hypertext links, ignoring forms [7] and client-side scripting. The web content behind forms and client-side scripting is referred to as the hidden web, which is estimated to comprise several millions of pages. Although there has been extensive research on crawling and exposing the data behind forms, [8] crawling the hidden web is induced as a result of client-side scripting. Crawling AJAX-based applications is fundamentally more difficult than crawling classical multi-page web applications. In traditional web applications, states are explicit and correspond to pages that have a unique URL assigned to them. In AJAX applications, the state of the user interface is determined dynamically through changes in the DOM that are only visible after executing the corresponding JAVASCRIPT code.

## 2. LITERATURE SURVEY

Modern web interfaces incorporate client-side scripting and user interface manipulation which is increasingly separated from server-side application logic. Although the field of rich web interface testing is mainly unexplored, much knowledge may be derived from two closely related fields traditional web testing and GUI application testing.

a) Authors Ali Mesbah and Arie van Deursen [1] [2][3] have suggested model for performining automatic testing of web application through invariant. It is based on invariants which does not give better result. Also it is more vulnerable for various attacks and does not provide mechanism for it.

b) Benedikt et al. [4] have presented a model for automatically exploring paths of multipage websites through a crawler and detector for abnormalities such as navigation and page errors (which are configurable through plugins). This model uses smart profiles to extract candidate input values for form-based pages. Although the crawling algorithm has some support for client-side scripting execution and provides insufficient detail to determine whether it would be able to deal with modern AJAX web applications

c) Author Y.W. Huang, C.H. Tsai, T.P. Lin, S.K. Huang, D.T. Lee, and S.Y. Kuo [9] have been proposed for automatically assessing web application security. The general approach is based on a crawler capable of detecting data entry points which can be seen as possible points of security attack. Malicious patterns, e.g., SQL and XSS vulnerabilities, are then injected into these entry points and the response from the server is analyzed to determine vulnerable parts of the web application

d) A model-based testing approach for web applications is proposed by Ricca and Tonella [10]. They introduced ReWeb, a tool for creating a model of the web application in UML, which is used along with defined coverage criteria to generate test cases, which rely on a finite state machine together with constraints defined by the tester. All such model-based testing techniques focus on classical multipage web applications. They mostly use a crawler to infer a navigational model of the web. Unfortunately, traditional web crawlers are not able to crawl AJAX applications.

## 3. COMMENTS

From the above survey we can comment that,

- In traditional web applications, states are explicit and correspond to pages having a unique URL (Uniform Recourse Locator).
- It does not handle browsers DOM tree.
- Reloading of whole page is occurred instead of reloading of specific content on the page. so it takes more times for generating response.
- Response to a client-side event can be injected into the single-page interface and therefore faults are propagating at the DOM level.
- It is more vulnerable to possible attacks.

## 4. NEED OF WORK

The work carried out in literature survey is based on traditional web application which having number of limitations. To overcome these limitations, we proposed automatic testing of web application by implementing AJAX crawler. In this AJAX crawler crawl the requested web page through the embedded web browser and find out clickable element before and after firing event on the clickable element. Based on the result we have generated DOM structure and compare it to analyze proposed system.

## 5. PROPOSED WORK

Here focus is to implement the AJAX Crawler and event generation for testing of modern web application which solves the boundaries of the traditional web application. Emphasis is given on analyzing browsers DOM tree for finding out clickable element on the application and generating different events on them for representing state flow graph of application. This provides all the possible transitions between different user interface states.

## 6. SYSTEM ARCHITECTURE

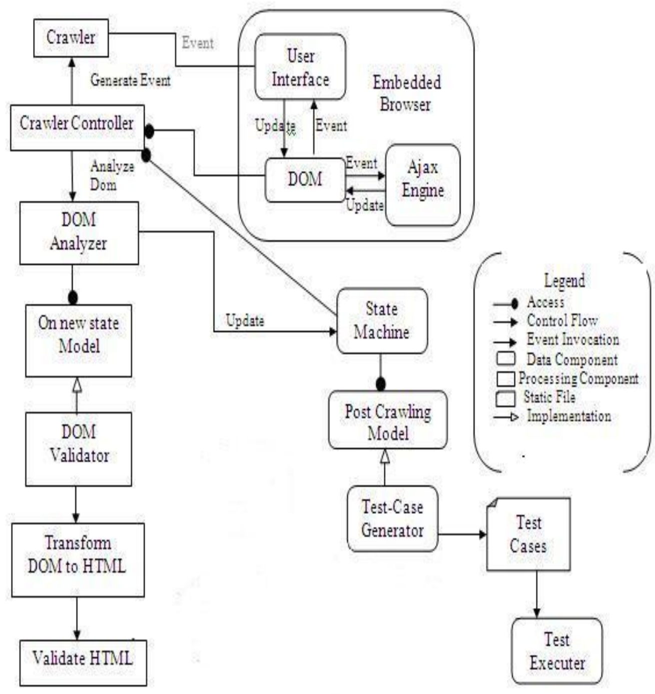


**Figure 1: System Architecture**

Figure 1 shows architecture of proposed system, in which crawler crawls the web pages that are to be opened through embedded browser. The crawler controller controls the execution of AJAX crawler by using different plug-ins with access of state machine, and based on the plug-ins execution DOM structure generated with validation.

## 7. THE MODULES OF THE PROPOSED WORK

### 7.1 Devising a mechanism to detect clickable element from web applications:

A crawling is more efficient for finding out clickable element from web application. The idea is to explore all the possible elements from web applications and those unexplored elements that are terminated left. We provide backtracking technique which is responsible for exploring unexplored element.

### 7.2 Devising a mechanism to find out the states and possible transitions before firing event on clickable element:

In this module we can open web application in the browser and analyze browsers DOM tree before firing event. This DOM tree gives states of user and possible transitions.

### 7.3 Devising a mechanism to find out the states and possible transitions after firing event on clickable elements:

In this module we can analyze browsers DOM tree after firing event and represent states and possible Transitions between them and infere a state flow graph for all possible transitions.

### 7.4 Devising a mechanism for analyze proposed system:

In this module we can provide new plugins for DOM validation and post-crawling for spotting out development errors and generation of validation report for web applications and compare the results.

## 8. EXPERIMENTAL SETUP

The proposed work is carried out by using Eclipse Kepler IDE with supported JDK 1.6 on Windows XP operating system. Initially we have used Eclipse Hellos with JDK 1.5 but, it is not supported for advanced libraries. We have configured AJAX application by adding external libraries that are to be supported for execution. To generate DOM structure of resulted web page, we have used DOM viewer and for comparison between two DOM structures used Wincompare tool is used.

## 9. RESULTS

Here, we have shown the result of proposed work, which implement a crawler for dynamic analysis of user interface state change on the given web application. In this, we have compared traditional web application with AJAX web application and it shows that AJAX web application is faster than traditional web application and also AJAX application having more functionality as compared to the traditional application.

**Table 1. Experimental benchmarks with different parameters**

| **Sr. No.** | **Attribute** | **Traditional Application** | **AJAX Application** |
|---|---|---|---|
| 1 | user request | synchronous execution | asynchronous execution |
| 2 | content | not updated dynamically | dynamically updated |
| 3 | response | “click, wait and page refresh” | no more “click, wait and page refresh”, |
| 4 | code | plain html | Javascript/Dhtml |

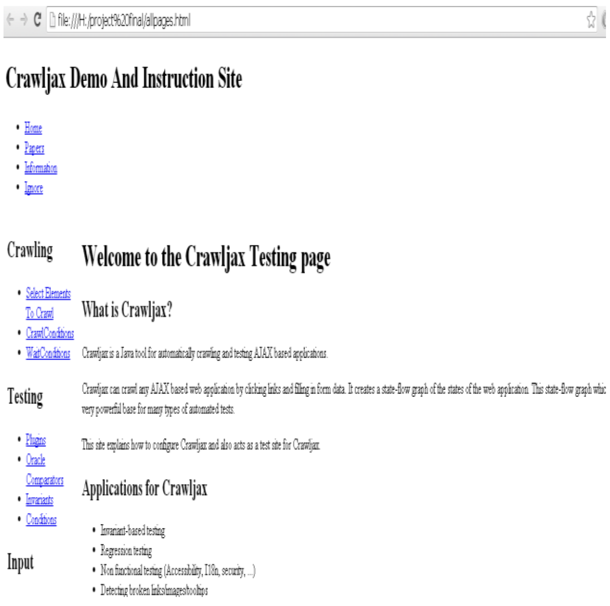
Table 1 indicates comparison analysis between traditional web application and AJAX web application by considering different parameters.

Work is carried out on each and every module separately. The step by step execution of work is mentioned below.

Step 1:

- Open the web application through the embedded browser.

- After opening the web page crawler is to be initialized to crawl the requested web page.



file:///H:/project%20final/allpages.html

Crawljax Demo And Instruction Site

- Home
- Papers
- Information
- Ignore

Crawling

- Select Elements To Crawl
- CrawlConditions
- WaitConditions

Testing

- Plugins
- Oracle Comparators
- Invariants
- Conditions

Input

Welcome to the Crawljax Testing page

What is Crawljax?

Crawljax is a Java tool for automatically crawling and testing AJAX based applications.

Crawljax can crawl any AJAX based web application by clicking links and filling in form data. It creates a state-flow graph of the states of the web application. This state-flow graph whic
very powerful base for many types of automated tests.

This site explains how to configure Crawljax and also acts as a test site for Crawljax.

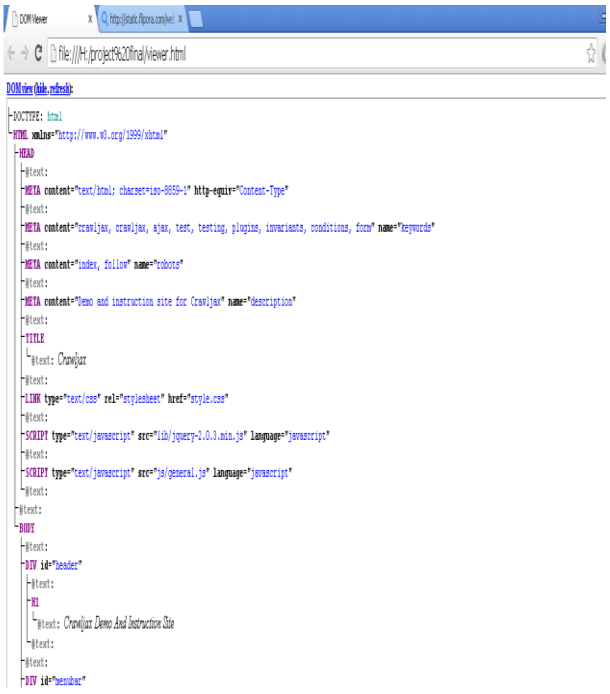
Applications for Crawljax

- Invariant-based testing
- Regression testing
- Non functional testing (Accessibility, I18n, security, ...)
- Detecting broken links/images/tooltips

**Figure 2: Loading First Web Page**

Step 2:

- After loading the web page crawler start to crawls the web page and find outs clickable elements and generate DOM tree.



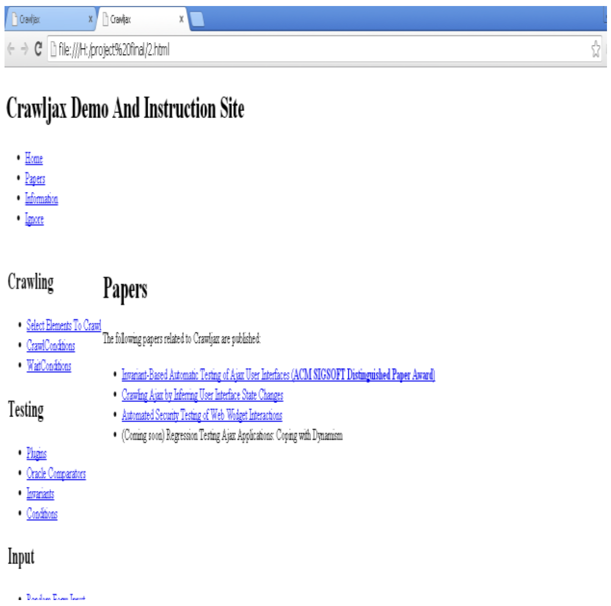
file:///H:/project%20final/viewer.html

```
├DOCTYPE: html
└HTML xmlns="http://www.w3.org/1999/xhtml"
  ├HEAD
  │ ├#text:
  │ ├META content="text/html; charset=iso-8859-1" http-equiv="Content-Type"
  │ ├#text:
  │ ├META content="crawljax, crawljax, ajax, test, testing, plugins, invariants, conditions, form" name="keywords"
  │ ├#text:
  │ ├META content="index, follow" name="robots"
  │ ├#text:
  │ ├META content="Demo and instruction site for Crawljax" name="description"
  │ ├#text:
  │ ├TITLE
  │ │ └#text: Crawljax
  │ ├#text:
  │ ├LINK type="text/css" rel="stylesheet" href="style.css"
  │ ├#text:
  │ ├SCRIPT type="text/javascript" src="lib/jquery-2.0.3.min.js" language="javascript"
  │ ├#text:
  │ ├SCRIPT type="text/javascript" src="js/general.js" language="javascript"
  │ └#text:
  ├#text:
  └BODY
    ├#text:
    ├DIV id="header"
    │ ├#text:
    │ ├H1
    │ │ └#text: Crawljax Demo And Instruction Site
    │ └#text:
    ├#text:
    ├DIV id="menubar"
```

**Figure 3: DOM Structure**

Step 3:

- In third step crawler again load the web page and find outs clickable elements and generate DOM tree.



file:///H:/project%20final/2.html

Crawljax Demo And Instruction Site

- Home
- Papers
- Information
- Ignore

Crawling

- Select Elements To Crawl
- CrawlConditions
- WaitConditions

Testing

- Plugins
- Oracle Comparators
- Invariants
- Conditions

Input

Papers

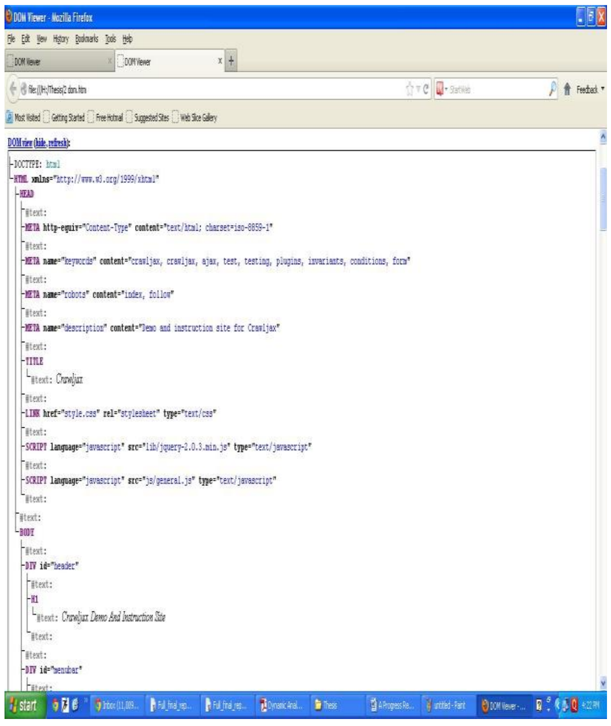
The following papers related to Crawljax are published:

- Invariant-Based Automatic Testing of Ajax User Interfaces (ACM SIGSOFT Distinguished Paper Award)
- Crawling Ajax by Inferring User Interface State Changes
- Automated Security Testing of Web Widget Interactions
- (Coming soon) Regression Testing Ajax Applications: Coping with Dynamism

**Figure 4: Loading Second Web Page**

Step 4:

- In fourth after loading the second web page crawler start to find out clickable elements.



file:///H:/Thesis/2 dom.htm

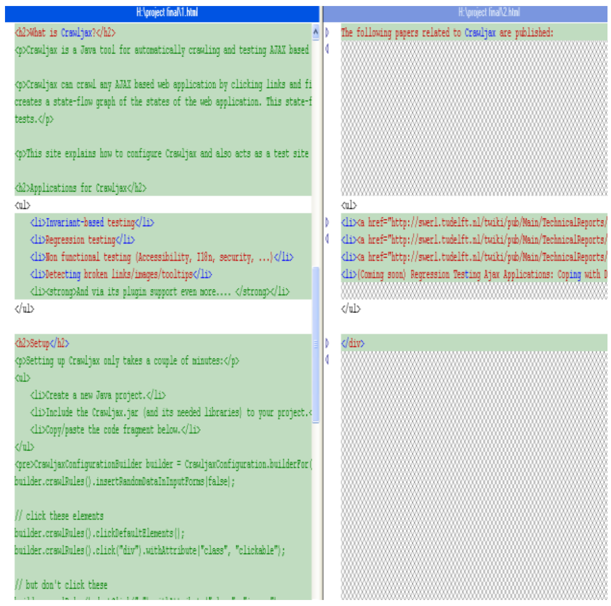
DOM view (hide, refresh):

```
├DOCTYPE: html
└HTML xmlns="http://www.w3.org/1999/xhtml"
  ├HEAD
  │ ├#text:
  │ ├META http-equiv="Content-Type" content="text/html; charset=iso-8859-1"
  │ ├#text:
  │ ├META name="keywords" content="crawljax, crawljax, ajax, test, testing, plugins, invariants, conditions, form"
  │ ├#text:
  │ ├META name="robots" content="index, follow"
  │ ├#text:
  │ ├META name="description" content="Demo and instruction site for Crawljax"
  │ ├#text:
  │ ├TITLE
  │ │ └#text: Crawljax
  │ ├#text:
  │ ├LINK href="style.css" rel="stylesheet" type="text/css"
  │ ├#text:
  │ ├SCRIPT language="javascript" src="lib/jquery-2.0.3.min.js" type="text/javascript"
  │ ├#text:
  │ ├SCRIPT language="javascript" src="js/general.js" type="text/javascript"
  │ └#text:
  ├#text:
  └BODY
    ├#text:
    ├DIV id="header"
    │ ├#text:
    │ ├H1
    │ │ └#text: Crawljax Demo And Instruction Site
    │ └#text:
    ├#text:
    ├DIV id="menubar"
```

**Figure 5: DOM Structure**

Step 5:

- We compare the resulted DOM tree of first web page and second web page and based on the comparison generate the final result.



H:\project final\1.html

```
<h2>What is Crawljax?</h2>
<p>Crawljax is a Java tool for automatically crawling and testing AJAX based

<p>Crawljax can crawl any AJAX based web application by clicking links and fi
creates a state-flow graph of the states of the web application. This state-f
tests.</p>

<p>This site explains how to configure Crawljax and also acts as a test site

<h2>Applications for Crawljax</h2>
<ul>
    <li>Invariant-based testing</li>
    <li>Regression testing</li>
    <li>Non functional testing (Accessibility, I18n, security, ...)</li>
    <li>Detecting broken links/images/tooltips</li>
    <li><strong>And via its plugin support even more.... </strong></li>
</ul>

<h2>Setup</h2>
<p>Setting up Crawljax only takes a couple of minutes:</p>
<ul>
    <li>Create a new Java project.</li>
    <li>Include the Crawljax.jar (and its needed libraries) to your project.<
    <li>Copy/paste the code fragment below.</li>
</ul>
<pre>CrawljaxConfigurationBuilder builder = CrawljaxConfiguration.builderFor(
builder.crawlRules().insertRandomDataInInputForms(false);

// click these elements
builder.crawlRules().clickDefaultElements();
builder.crawlRules().click("div").withAttribute("class", "clickable");

// but don't click these
```

H:\project final\2.html

```
The following papers related to Crawljax are published:

<ul>
<li><a href="http://swerl.tudelft.nl/twiki/pub/Main/TechnicalReports/
<li><a href="http://swerl.tudelft.nl/twiki/pub/Main/TechnicalReports/
<li><a href="http://swerl.tudelft.nl/twiki/pub/Main/TechnicalReports/
<li>(Coming soon) Regression Testing Ajax Applications: Coping with D
</ul>

</div>
```

**Figure 5: DOM Comparison**

It shows the result of DOM structure before executing event on clickable element and after executing event on clickable element. Here we clearly identified changes made in the browsers DOM structure.

## 10. CONCLUSION

In this paper, we have proposed a method for testing AJAX application automatically. Our current work consists extending the crawler functionality for supporting automated testing of modern web applications. We implement the plugins for automatic testing. Our future work will include development of more testing plug-ins.

## 11. ACKNOWLEDGEMENT

I would like to thanks my guide Prof.A.B.Rajmane for his valuable and constructive comments. I would also like to thanks Prof. Mrs. A.N. Mulla, HOD Computer Science & Engineering Department, Annasaheb Dange College of Engineering & Technology, Ashta for her valuable support.

## 12. REFERENCES

[1] Ali Mesbah, Member, IEEE Computer Society,Arie van Deursen, Member, IEEE Computer Society, and Danny Roest " Invariant-Based Automatic Testing of Modern Web Applications" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 38, NO. 1, JANUARY/FEBRUARY 2012.

[2] A. Mesbah and A. van Deursen, "Invariant-Based Automatic Testing of Ajax User Interfaces," Proc. IEEE 31st Int'l Conf. Software Eng., pp. 210-220, 2009.

[3] A. Mesbah, E. Bozdag, and A. van Deursen, "Crawling Ajax by Inferring User Interface State Changes," Proc. Eighth Int'l Conf. WebEng., pp. 122-134, 2008.

[4] M. Benedikt, J. Freire, and P. Godefroid, "VeriWeb: Automatically Testing Dynamic WebSites," Proc. 11th Int'l Conf. World Wide Web, pp. 654-668, 2002.

[5] Ali Mesbah, Arie van Deursen, and Stefan Lenselink "Crawling Ajax-based Web Applications through Dynamic Analysis of User Interface State Changes" Report TUD-SERG-2011-033.

[6] Cristian Duda , Gianni Frey, Donald Kossmann , Reto Matter, Chong Zhou ,

ETH Zurich, Switzerland "AJAX Crawl: Making AJAX Applications Searchable" IEEE International Conference on Data Engineering 2009. Web Information and Data Management (WIDM'04). ACM Press, New York, NY, 9–15.

[8] LAGE, J. P., DA SILVA, A. S., GOLGHER, P. B., AND LAENDER, A. H. F. 2004. Automatic generation of agents for collecting hidden Webpages for data extraction. Data Knowl. Eng. 49, 2, 177–196.

[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] 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.