UNIVERSITY OF GOTHENBURG

Search-based testing tools for Ajax

A systematic literature review

Bachelor of Science Thesis in Software Engineering and Management

MARKUS FEYH

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, August 2013

Search-based testing tools for Ajax
A systematic literature review

Markus Feyh

Examiner: Morgan Ericsson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden August 2013

Search-based testing tools for Ajax - a systematic literature review

Markus Feyh
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
Email: marfeyh@gmail.com

***Abstract*—*Background*: Search-based testing seeks to solve many novel problems including testing Ajax applications, and there have been a number of tools created to accomplish this purpose.**

***Objective*: This thesis aims to identify search-based software testing tools for Ajax web applications and how they have been evaluated.**

***Method*: Systematic literature review is used as the research methodology.**

***Result*: There are six different tools identified in scientific literature of which three are variants of Crawljax. Also, search-based testing tools for Ajax are primarily evaluated through the use of case studies.**

***Conclusion*: The evaluation of the identified tools should be conducted using an experimental design in order to make them comparable and repeatable as well as follow benchmarking frameworks proposed in scientific literature.**

***Index Terms*—Ajax, search based testing, systematic literature review, tools.**

I. Introduction

What is the best way to test Ajax web applications? Search based software engineering (SBSE) is regarded as a solution to many of software engineering's problems. SBSE proposes that by posing each problem as a search-based optimization problem, an optimized solution to the problem may be found [1]. As a result of the potential to tackle the novel problems of Ajax web applications through the use of search-based software testing (SBST) on Ajax web applications has grown to include a body of knowledge worthy of review.

Web technologies play an important role. One of the web technologies, Ajax, the combination of asynchronous JavaSript and XML, is the area of focus in this paper. This grouping of web technologies requires novel testing approaches which present new issues but are important to test [2]. With Ajax, the document object model (DOM) structure of the website can partially change as a result of asynchronous messaging.

Ajax's features present a challenge in comparison with traditional web technologies because of the added complexity of asynchronous communication and partial DOM structure changes. Given that testing should be optimized, it is important to research this area in order to understand the benefits of using novel methods such as search based testing. Additionally, the different features of Ajax in comparison with traditional web technologies require novel testing approaches, which through experimentation can be shown to be more or less effective.

This thesis project's aim is to understand the search-based techniques for testing Ajax web applications. Ajax contains features which make it challenging to test and search-based software testing offers a solution to novel problems. Thus, the motivation for undertaking this thesis was to understand which tools have been applied from the search-based testing domain to Ajax web applications. In order to do this, a Systematic Literature Review [3] was undertaken in order to understand the state-of-the-art.

II. Background

A. Search-based Software Engineering

The field of search-based software engineering seeks to transform problems into optimization search problems. This kind of searching differentiates itself from textual or hypertextual searching. A search-problem is , "(...) in which optimal or near optimal solutions are sought in a search space of candidate solutions, guided by a fitness function that distinguishes between better and worse solutions." [1]. Using these search-problems as the optimization problems, SBSE focuses on solving them in an optimal way. Even if the most optimal solution is not achieved, a near optimal solution is.

Relevant to this thesis is the field of testing which falls under search-based software engineering. This field is defined as search-based testing, which is one of many fields which search-based software engineering has been applied to.

B. Search-based Software Testing

This thesis falls under the broad area of search-based software testing, but differentiates itself by focusing on Ajax web applications. The field of search-based testing has been systematically review by Afzal *et al.* [4]. In their systematic literature review it is defined as, "Search-based software testing (SBST) is the application of metaheuristic search techniques to generate software tests. The test adequacy criterion is transformed into a fitness function and a set of solutions in the search space are evaluated with respect to the fitness function using a metaheuristic search technique" [4]. Thus, one can infer that search-based testing is based on the following three components:

- metaheuristic search techniques,
- fitness functions, and
- test suite generation.

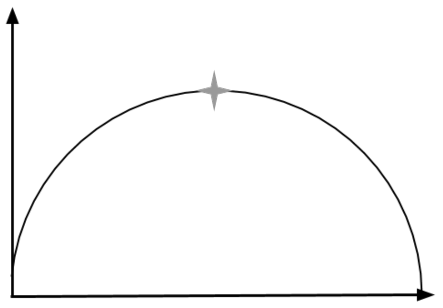

Fig. 1. Optimization using a local search metaheuristic.

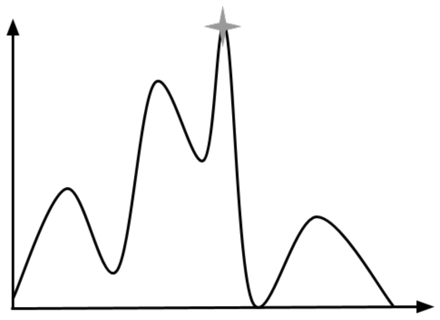

Fig. 2. Optimization using a global search metaheuristic

Each of these three components are necessary in order for a tool to be considered search-based in the field of testing. The final component of test suite generation is when test cases are generated. Test cases execute the program in a specified way in order to test a desired functionality. Test suites consist of the aforementioned test cases. In the following sections, metaheuristic search techniques and their fitness functions are distinct, and need further elaboration.

1) Metaheuristic Search Techniques: The term 'search' in search-based software testing refers to the fact that a metaheuristic is used. A metaheuristic is defined by Osman and Laporte [5] as being, "(...) an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space (...)". In other words, metaheuristics are algorithms which seek to optimize a problem. They usually do not lead to deterministic answers. Instead, metaheuristics are generally non-deterministic. This means that they seek a solution near the most correct one, but may not reach a perfectly correct solution.

Two examples of metaheuristics are global and local search algorithms. These algorithms are preferred in different situations. The difference between local and global search algorithms represent one of many differentiations between search algorithms. Because there are a plethora of algorithms which can be used to search for the most optimized solution, domain knowledge is needed in order to select the correct one.

Local search algorithms are efficient in finding single global maximums. For example, a local search algorithm such as hill climbing [6] would be suitable to Fig. 1. By using the algorithm, any candidate solution along the graph would first be chosen, and by checking the neighboring solutions the algorithm would be able to quickly locate the maximum. The local search algorithm is applicable in this specific situation. However, in situations where the global maximum is being sought it is inefficient if there are many local maxima.

A solution to the limitations of the local search algorithm are global search algorithms. An example of this kind of algorithm is simulated annealing. Simulated annealing [7] slowly cools allowing probabilistic jumping to non-optimal candidate solutions throughout the graph. Over time, the algorithm will settle into a local maximum in situations such as Fig. 2 more consistently. However, the simulated annealing algorithm is less efficient then hill climbing in the previous example seen in Fig. 1. Each metaheuristic presents a different set of trade-offs, so the correct selection of the algorithm to match the situations is important.

Metaheursitics are one of three vital components in search-based software testing. Furthermore, the selection of correct algorithm is important based on the problem which is being optimized.

2) Fitness Function: The metaheuristic used is guided by a fitness function. A fitness function, "distinguishes between better and worse solutions" [1]. In other words, it decides whether candidate solutions tested by the metaheuristic are more or less fit. As a result, a more optimal solution is reached that meets the needs of the fitness functions. For example, in Fig. 1 and Fig. 2 the fitness function is defined as finding the maximum y coordinate of the graphed functions.

C. Ajax

Ajax is a grouping of technologies which are used web applications. The term itself was pioneered by Garrett [8] and defined as consisting of:

- "standards-based presentation using XHTML and CSS;
- dynamic display and interaction using the Document Object Model;
- data interchange and manipulation using XML and XSLT;
- asynchronous data retrieval using XMLHttpRequest;
- and JavaScript binding everything together". [8]

These features together comprise what is collectively defined as Ajax. Of note, is that Ajax has evolved over time in order to include a broader definition including web applications even programmed using HTML instead of XHTML and JSON instead of XMLHttpRequest.

There are two features of Ajax, which make it particularly difficult to test. The first is its use of asynchronous communication. Secondly, is its ability to dynamically update web pages partially.

Asynchronous communication is when data is sent between the client and the server without it's receipt being confirmed. Furthermore, the order in which data is sent using Ajax can be mixed, and so it presents a number of complicated challenges to testing. For example, when a test suite is written it does

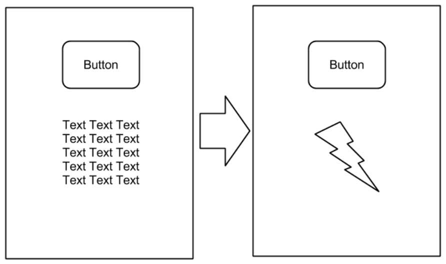

Fig. 3. Dynamic Updating

whether a certain functionality works. However, in order that it cope with asynchronous communication, the possibility that communication will be sent or received out of order is a key consideration. While testing, asynchronous communication is an added concern.

The second issue is the use of dynamically updated web pages. A dynamically updated web page can have its document model object (DOM) partially updated without changing the current web page. The DOM consists of the web pages HTML, XHTML or XML structure which are presented to the user. For example, in Fig. 3 it may be seen that when an action occurs such as the button is pushed, then the text will dynamically be updated to display an image instead. This functionality is useful since it allows only a part of the DOM structure to be updated without having to update the whole page.

Dynamically updating web pages presents a challenge in testing [2] since traditional web pages are fully reloaded when the DOM is changed. However, tools which test Ajax need to cope with this additional functionality of the technology.

III. Related Research

There is little research in the field of dynamically understanding programs [9]. An area of research that falls under the field of dynamically understanding programs is reverse engineering. Reverse engineering is one approach for coping with Ajax's asynchronous messages and DOM changes. The goal of dynamic reverse engineering is to understand the website's states. Furthermore, in the context of web technologies, the larger field of reverse engineering has been reviewed by [10]. However, the review focused on more traditional web technologies, and did not address Ajax. The current state of the field is limited in terms of understanding, and research has yet to systematically understand how to cope with Ajax.

Currently, there are a number of approaches which have been proposed and studied in research in the context of dynamically understanding and testing Ajax. The first is FireDetective [11], a tool that uses traces from both the client and server side to dynamically understand how the website works. The tool has been shown in a small user study to improve effectiveness, efficiency and confidence of developers [12] when working with Ajax websites. Secondly, Crawljax [13] is a tool that uses a novel approach for crawling a website that copes with the differences of Ajax. Using an algorithm it identifies and automatically constructs a state machine of the website. The tool has been presented in a number of empirical studies where it has been applied. Thirdly, an agile approach which incrementally reverse engineers Ajax user interfaces named CReRia [14] is presented in a preliminary experiment. The tool facilitates iterative reverse engineering of the finite state machine of the interface through clustering components based on heuristic criteria. Finally, ReAjax [15] is a tool that uses dynamic traces and focuses on the client-side perspective. Additionally, it focuses on single Ajax web pages by identifying indicators and mutators on a single web page before collecting traces. Marchetto and Tonella [15] claim that this leads to a richer finite state machine, which would be missed by the other tools.

As a result of research into the dynamic reverse engineering of Ajax, some of the tools have been applied in the field of testing as well. The most widely used is ATUSA, a plug-in for Crawljax. It has been applied in the field of testing for automatic invariant [13], security [16], regression [17] and cross-browser compatibility testing [18]. ReAjax has been used for testing purposes as well. It been used for testing both semantically interacting events [19] as well as optimization of this technique using search based testing [15]. Finally, FireDetective and CReRia focus primarily on increasing understanding of web applications, and have not been used as widely in the context of testing.

A comparison of CReRia, ReAjax and Crawljax has previously occurred [20] based on the comparative framework defined by Gueheneuc *et al.* [21]. The comparison was done at a high level which was focused on a broad set of criteria. Furthermore, a systematic literature review [22] was undertaken to understand the status of how SBST is used for test-case generation and also provides a framework for how tools should be empirically evaluated.

IV. Research Methodology

A. Research Questions

RQ1: What are the state-of-the-art tools for search based testing of Ajax applications?

RQ2: How have search-based testing tools been evaluated in scientific papers?

B. Systematic Literature Review

The guidelines for conducting the systematic literature review (SLR) come from Kitchenham [3]. An SLR is useful when a body of scientific literature needs to be systematically identified and information extracted. A benefit of structuring it follow a set of guidelines is that it is both understandable and repeatable by others. By conducting the SLR, the relevant literature will be used to identify the state-of-the-art tools and how they have been evaluated in order to answer **RQ1**.

The terminology for the SLR and design of the review protocol come from the relevant sections of the SLR guideline [3]. This section details the steps taken to conduct the SLR.

The population, intervention, outcome, context and experimental design defined in this section come from the SLR guidelines [3]. Population for this study is the field of testing of

Ajax applications. The intervention is the use of search based software engineering to detect defects in the Ajax applications. Outcome is the varying ability of defect detection of search based engineering approaches. The context is restricted to Ajax applications. Finally, the experimental design is not restricted.

Next, in order to answer **RQ2** the methodology which each paper uses for evaluation of the search-based tool and a summary of results will be presented in a table. This will be done in order to visualize how the tools have been evaluated for the reader.

An overall view of the research methodology can be found in Fig. 4. The rectangle represent objects or outcomes, while the arrows represent actions. Finally, the circles listing research question numbers identify the outcome which will answer them.

C. *Selection Criteria*

V. Search Strategy

The search strategy focuses on finding papers relevant to the population, intervention and outcome. [3] The key terminology was combined into a single search string. No synonyms were identified, because the terms were broad enough. The operator AND was used to join the major terms. The search strings were chosen in order to answer the research questions of the SLR. Overall, the SLR used a search-term based strategy for study discovery.

A pilot search was conducted and it was found that using the synonym 'crawl*' of 'search*' resulted in a broader set of search-based studies. Using the synonym helped identify the important literature better. The resulting body of research was found to be an appropriate size for the area.

The search terms that were used were based on the following population, intervention and outcome:

- Population: testing.
- Intervention: search, crawl.
- Outcome: Ajax.

The author searched for journal and conference articles using SCOPUS[1], and the Inpsec and Compendex[2] databases. They were chosen, because they contain peer-reviewed journal and conference articles from many content providers. Thus, it was possible to both have peer-reviewed research and a wide array of journals and conferences to search in.

The search was not limited to any field or time period in order to see what was available. The following search queries were used:

SCOPUS: *TITLE-ABS-KEY((search* OR crawl*) AND test* AND ajax)*

Compendex & Inspec: *((search* OR crawl*) AND test* AND ajax WN KY)*

[1]www.scopus.com
[2]www.engineeringvillage.com

During the SLR, only journal and conference articles were considered. They were considered to be more reliable and representative of a higher quality of research.

For each query, the author took all returned papers into consideration. The articles were then used for conducting the SLR. Each paper considered also needed an author. For example, the proceedings of a conference were not considered. For any duplicate papers were found, they were skipped and not taken into account.

In order to identify the relevant papers for understanding search-based approaches to testing Ajax applications as well their effectiveness in discovering defects the author made a checklist. Each paper was selected based on the following questions:

- Does the paper focus on testing Ajax applications?
- Does the paper describe an approach for search-based testing?
- Does the paper describe whether the approach is effective?
- Does the paper provide evidence of defect detection?

Every question was scored using the abstract of each research paper. Papers with a score greater than 2.5 were included. Each question was assessed using the following criteria:

- Full = 1 point
- Partial = 0.5 points
- None = 0 points

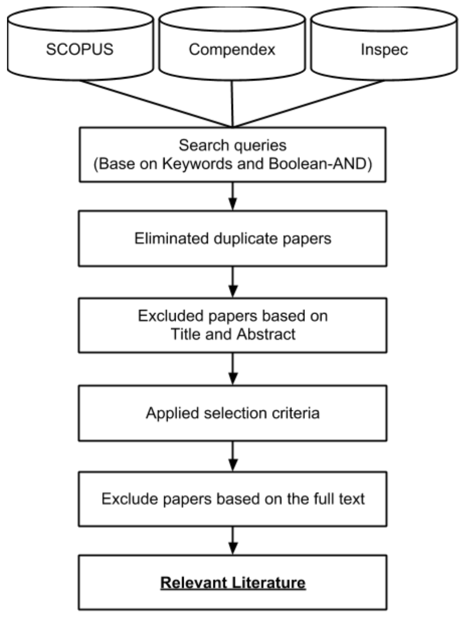
SCOPUS
Compendex
Inspec
Search queries
(Base on Keywords and Boolean-AND)
Eliminated duplicate papers
Excluded papers based on
Title and Abstract
Applied selection criteria
Exclude papers based on the full text
Relevant Literature

Fig. 5. Search Strategy.

A. *Quality Assessment*

The quality of each selected article was reviewed for quality based on the following seven questions:

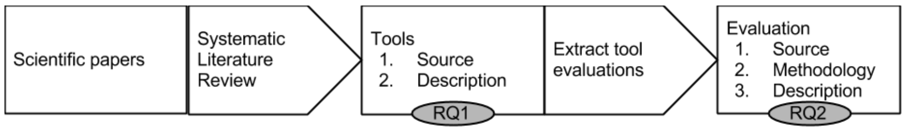

Fig. 4. Summary of research methodology.

TABLE I
RELEVANT LITERATURE

Paper Title
Automated acceptance testing of JavaScript web applications [23]
Automated acceptance testing of JavaScript web applications [23]
Automatic AJAX application testing [24]
Crawling AJAX-based web applications through dynamic analysis of user interface state changes [13]
Crawling AJAX by inferring user interface state changes [25]
Invariant-based automatic testing of Ajax user interfaces [26]
Invariant-Based Automatic Testing of Modern Web Applications [27]
Regression Testing Ajax Applications: Coping with Dynamism [17]
Search-based testing of Ajax web applications [19]
Using search-based algorithms for Ajax event sequence generation during testing [15]

- Is the goal of the research understandable?
- Is the research methodology adequately described?
- Are threats to validity identified?
- Are limitations of the research discussed?
- Does the paper provide results that are repeatable?
- Do the conclusions relate to the research?
- Does the paper discuss areas of future research?

By asking the previously stated questions to each article, the articles which are of a high quality can be determined.

B. Data Extraction Process

To perform the relevant article search, the authors inserted the search query in the SCOPUS and the Inpsec and Compendex database. The databases produced 48 articles in total. After eliminating duplicate papers and articles without authors, such as conference proceedings, there were in 17 papers in total that resulted from the search strategy. These 17 papers were then accessed by the author against the selection criteria. Papers needed a relevance score of 2.5 points in order to be considered in the SLR. There were 9 relevant papers that resulted from the literature after performing the selection scoring. The process can be seen in Fig. 5. The relevant literature will be the focus of this SLR.

All relevant literature was imported into Zotero [3]. A listing of relevant articles may be found in Table 1.

VI. ARTICLE QUALITY

The quality of each article was assessed in order to present the state of research in the context of its quality. The following questions were asked for each article:

[3]http://www.zotero.org/

TABLE II
QUALITY SCORES OF PAPERS

Points	3.0-3.5	4.0-4.5	5-5.5	6-6.5
Papers	[23], [24]	[25]	[15], [19]	[13], [26], [27]

- Is the goal of the research understandable?
- Is the research methodology adequately described?
- Are threats to validity identified?
- Are limitations of the research discussed?
- Does the paper provide results that are repeatable?
- Do the conclusions relate to the research?
- Does the paper discuss areas of future research?

For each question, the full text of the study was consulted. Based on the fulfilment of the quality criteria the following scores were assigned for each question:

- Full = 1 point
- Partial = 0.5 points
- None = 0 points

The total points of each paper is summarized in Table 2. All papers presented understandable goals, discussed future research and had conclusions that related to the papers research. Where the quality scored lowest was in regards to whether the research methodology was adequately described. When a research methodology was described, the authors cited Yin [28] without providing much further detail. Other issues of paper quality had to do with the lack of discussion around limitations of the research, although this could be considered a growing pain of recent research in a new area. Additionally, the results presented were not immediately repeatable for the author based on information provided in the paper. Overall, the quality of the papers seemed acceptable for the papers to be considered as part of future research.

VII. RESULTS

In order to answer the research questions the relevant information was extracted from each article. The relevant information is described below.

RQ1: What are the state-of-the-art tools for search based testing of Ajax applications?

There are a number of tools related to search-based testing Ajax applications. The first is Crawljax [25], which reverse engineers the web application through the use of crawling. It even can create a static version of the website based on captured states that can be used for testing. The author later presented the development of the tool [27] along with a number of empirical case studies of its use. The Crawljax tool uses an edit distance algorithm that can be considered

TABLE III
TOOL VARIANTS BASED ON CRAWLJAX.

Tool	Invariants	User-stories	Test suites
NDIRT	Yes	No	Limited
Crawlscripter	No	Yes	Limited
ATUSA	Yes	No	Yes

a search-based algorithm. The edit distance algorithm comes from Levenshtein [29] and the fitness function which optimizes the algorithm is a complex difference algorithm that checks the DOM structure for differences [30].

Although Crawljax does not fully test Ajax web applications, it has been extended in a plug-in named ATUSA. ATUSA [26] itself derives tests from the DOM state transitions that the Crawljax tool collects. The creation of the test suites is accomplished through the hooks that can be activated before, after and during crawling. Since ATUSA is built as a plug-in for Crawljax it can also be considered a tool for search-based testing of Ajax applications.

In an extension to Crawljax, a technique for dealing with non-deterministic invariant regression testing (NDIRT) was proposed [17]. This is applicable when regression testing is done, because there are new states added. When testing these new states there are a number of challenges that result from detecting them correctly. In order to deal with these challenges, the state comparison filters out specific parts of the DOM structure such as the date and time [17]. As a result, similar states that only differ in time can be considered similar, thereby making regression testing more robust.

In order to make Crawljax more intuitive, a high level interface that uses a human-readable scripting language was proposed name Crawlscripter [23]. The tool can be used by someone who is not familiar with programming in order to test an Ajax application. Through its human readable scripting language, user stories can be scripted more naturally.

Summarized in Table 2 are the tool variants of Crawljax which have been identified and their respective properties. Key differences are identified as the ability of the tool to handle invariants (see ”Invariants” column in Table 2), program user stories (see ”User-stories” column in Table 2) and the ability to generate test suites (see ”Test suites” column in Table 2).

ReAjax is a search-based tool [15], [19] that uses dynamic traces from Ajax applications in order to generate test cases. It focuses on single-page Ajax applications. The tool differentiates itself from other tools, because it requires manual instrumentation of the DOM structure. Also, it explicitly seeks to apply a search-based method of testing. This is accomplished through the generation of test suites based on algorithms that seek out the most optimal test cases based on an optimization function.

Another approach to testing Ajax applications was proposed in the form of an initial architectural description [24] of the tool. The proposed tool would represent the Ajax application as number of states which it would use to generate test cases.

TABLE IV
SEARCH-BASED TESTING TOOLS.

Tool	Metaheuristic	Fitness function	Test suites
Crawljax	Levenshtein [29]	Diff [30]	Limited
NDIRT	Levenshtein [29]	Diff [30]	Limited
Crawlscripter	Levenshtein [29]	Diff [30]	Limited
ATUSA	Levenshtein [29]	Diff [30]	Yes
ReAjax	Multiple	Multiple	Yes
STS	STS Algorithm [31]	Pre-defined set	Limited

In order to decide on the correct number of test cases generated to test state transitions, heuristics would be used. Overall, the concept is only presented in an initial architecture with a Unified Modeling Language (UML) to Symbolic Transition System (STS) converter created. In this thesis the tool will be referred to as STS .

In Table 3, a summary of the presented tools is shown based on the definition of search-based testing. Search-based testing is defined as a testing approach that has the following three components: a metaheuristic, fitness function and test suite generation. As seen from the table all tools have these elements. However, the ability for Crawljax, and its variants (see Table 2) to generate test suites is limited to a smoke test level of complexity so it is identified as limited. This is in contrast to the ability of ATUSA and ReAjax, which can generate test suites in many ways. For example, ReAjax can change fitness functions and ATUSA has the ability to define pre-, during and post- hook actions while crawling.

RQ2: How have search-based testing tools been evaluated in scientific papers?

Crawljax is assessed from the perspectives of accuracy that the results are correct, scalability that the tool is applicable in the real world, and the performance of the application itself [13]. In order to evaluate these three perspectives, six different Ajax applications were crawled. The accuracy was assessed by comparing the number of expected clickables to the actual clickables detected using the tool. The tool discovered almost all expected clickables for the six applications with only the exception of a few clickables that needed to click the same element multiple times in order to make them visible. Next, the scalability was assessed in terms how long the Ajax application crawling took. Overall, none of the six applications took longer than two hours. This showed that the scalability to large Ajax was possible. Finally, the applicability of tool in terms of running in multiple browsers was shown to have a positive effect on runtime when multiple browsers were used to test Google AdSense [4]. The author believes the tool is accurate, scalable and applicable in the real world. It has also been used for security testing and cross-browser compatibility testing. By running Crawljax itself, the tool can be considered a kind of smoke test as described by Memon [32].

ATUSA’s effectiveness [26] was evaluated in its code coverage and ability to discover seeded faults in the Ajax

[4]https://www.google.com/adsense/

application. In the first case study, the TuDu application[5] was tested. Out of ten seeded faults, eight were discovered. Additionally, there was 75% client-side and 73% server-side code coverage. Moreover, the time it took in order to run the tests was low, i.e. 26.5 minutes manual work and 6.5 minutes crawling.

The second case study using ATUSA [26] focused on an Ajax application for teachers[6] in order to understand the tools effectiveness during development. The developers used in the case study found it easy to define invariants in only a few lines of code. As a result of the tool being used, six faults were discovered that were not found through manual testing.

The regression testing (NDIRT) approach by Roest *et al.* [17] dealt with dynamic invariants encountered using Crawljax through the combination of the implemented comparator and element resolver. The comparator was a comparison function that imposed custom expectations on the DOM structure. Furthermore, the element resolver helped correctly identify state changes that would otherwise not have been identified correctly, thus leading to incorrect state changes. As a result, it was reported that this resulted in the avoidance of false negatives and false positives were reduced in some cases completely. Additionally, by using custom comparators, the number of false positives was reduced from 50 to 95% when testing the Google Reader Ajax application [7]. This is a significant sign of effectiveness since the Google Reader application is extremely dynamic.

The ReAjax tool was tested in two case studies and its performance was presented in terms of the combination of three factors: (i) bound or unbound test suite size, (ii) algorithm used and (iii) fitness function used. The algorithms performance in terms of computation time varied greatly. Performance time was clearly better for bounded test suites than for unbounded. Furthermore, the performance of fitness functions was varying based on the application tested. For simplicity, the most successful algorithm and fitness function results will be considered. In both case studies twenty faults were injected. In the TuDu application, the bounded suite detected 66.67% of injected faults, while the unbounded suites detected approximately 70%. For the Oryx application, the bounded test suite detected 84.6% of faults and the unbounded test suite detected 100% of injected faults. The authors found that even using bounded test suites, ReAjax is still reasonably effective.

Crawlscripter [23] was shown to be effective in implementing eight user stories from the JPetStore application[8]. Additionally, it was applied to check if citations could be downloaded through Springer Link[9]. Crawlscripter through these two case studies was shown as being effective in converting user stories into tests.

[5]http://tudu.sourceforge.net
[6]http://www.coachjezelf.nl
[7]http://www.google.com/reader
[8]http://java.sun.com/developer/technicalArticles/J2EE/petstore/
[9]http://www.springerlink.com

TABLE V
EVALUATION OF TOOLS.

Source	Tool	Methodology	Test subject
[15], [19]	ReAjax	Case study	Tudu, Oryx
[13], [25]	Crawljax	Case study [28]	JpetStore, Gucci, Internal, Sports center, Online shop
[13]	Crawljax	Case study	Google AdSense
[27]	ATUSA	Case study [28]	HitList, TheTunnel
[26], [27]	ATUSA	Case study [28]	TuDu
[26], [27]	ATUSA	Case study [28]	Coachjezelf
[17]	NDIRT	Case study [28]	HitList, Google Reader
[24]	STS	N/A	None
[24]	Crawlscripter	Case study	JpetStore

VIII. Threats to Validity

The most major threats to validity of this thesis is the possibility that only a portion of relevant scientific papers were selected as relevant literature, the possibility that the author's data extraction was subjective.

In the research methodology the author chose to conduct a Systematic Literature Review following the procedure from Kitchenham [3]. There is however a possibility that the procedure and protocol created was inadequate to capture the relevant scientific publications. The author added the search term 'crawl' in order to include more relevant research and expand the amount of papers considered. However, finding a search string that is comprehensive is a hard task.

The second major threat to validity is that the author may have extracted the data in a subjectively. This can be due to misunderstandings that occur during the extraction process. The author reviewed the data extracted, and checked if the information was correct in connection with the papers. The extraction of data was done only by the author and the analysis as well. The author has reviewed the information in thesis, but there is a threat that it could contain subjective information.

IX. Conclusions

As a result of conducting the Systematic Literature Review (SLR) on search-based testing of Ajax applications a number of tools were identified in order to answer RQ1. Those tools were: ReAjax [15], [19], Crawljax [13], [25], ATUSA [26] and Crawlscripter [23]. An additional paper [17] proposed improvements to Crawljax by evaluating a non-deterministic invariant regression testing (NDIRT) approach. Finally, a proposal for a tool that used symbolic transition system (STS) testing [24] was found, which has not been implemented yet. Both the tools Crawlscripter and ATUSA were based on Crawljax. ReAjax, existed as a self-contained tool, and was not based on any other tool contained in the SLR.

In summary, as a result of conducting the SLR on search-based testing tools for Ajax web applications the author found the following answers to **RQ1**:

- ReAjax, Crawljax, ATUSA and Crawlscripter identified as tools,

- ATUSA are Crawlscripter are based on Cralwjax,
- An STS approach was proposed but not implemented,
- NDIRT is an improvement on Crawljax.

As an answer to **RQ2** it was found that the tools were primarily evaluated through the use of case studies. These case studies tested Ajax web applications. However, the test results were not comparable since no case study design and procedure was the same. Further evaluation is needed in order to draw conclusions beyond the individual case studies each paper presents.

In summary, the results of **RQ2** can be understood to indicate:

- Case study was the primary methodology for evaluation of tools,
- No case study shared the same design and procedure.

Based on the tools discovered, the field of search-based testing of Ajax applications is a recent phenomenon since relevant scientific only dates back to 2008. Further research should be done in the area to evaluate tools in a way that is comparable. For example, by designing an experiment that multiple tools can be benchmarked against or following the framework proposed by Ali *et al.* [22] for evaluation the SBST tools.

References

[1] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," *ACM Comput. Surv.*, vol. 45, no. 1, p. 11, 2012.

[2] A. van Deursen and A. Mesbah, "Research issues in the automated testing of ajax applications," in *SOFSEM*, 2010, pp. 16–28.

[3] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, p. 2004, 2004.

[4] W. Afzal, R. Torkar, and R. Feldt, "A systematic review of search-based testing for non-functional system properties," *Information & Software Technology*, vol. 51, no. 6, pp. 957–976, 2009.

[5] I. H. Osman and G. Laporte, "Metaheuristics: A bibliography," *Annals of Operations Research*, vol. 63, no. 5, pp. 511–623, 1996.

[6] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*. Prentice hall Englewood Cliffs, 1995, vol. 74.

[7] B. Suman and P. Kumar, "A survey of simulated annealing as a tool for single and multiobjective optimization," *Journal of the operational research society*, vol. 57, no. 10, pp. 1143–1160, 2005.

[8] Jesse James Garrett, "Ajax: A new approach to web applications - adaptive path," http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications, Feb. 2005. [Online]. Available: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

[9] B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke, "A systematic survey of program comprehension through dynamic analysis," *IEEE Trans. Software Eng.*, vol. 35, no. 5, pp. 684–702, 2009.

[10] R. Martin and L. Archer, "Reverse engineering of web applications: A technical review," 2007.

[11] N. Matthijssen and A. Zaidman, "Firedetective: understanding ajax client/server interactions," in *ICSE*, 2011, pp. 998–1000.

[12] N. Matthijssen, A. Zaidman, M.-A. D. Storey, R. I. Bull, and A. van Deursen, "Connecting traces: Understanding client-server interactions in ajax applications," in *ICPC*, 2010, pp. 216–225.

[13] A. Mesbah, A. van Deursen, and S. Lenselink, "Crawling ajax-based web applications through dynamic analysis of user interface state changes," *TWEB*, vol. 6, no. 1, p. 3, 2012.

[14] D. Amalfitano, A. R. Fasolino, and P. Tramontana, "Rich internet application testing using execution trace data," in *ICST Workshops*, 2010, pp. 274–283.

[15] A. Marchetto and P. Tonella, "Using search-based algorithms for ajax event sequence generation during testing," *Empirical Software Engineering*, vol. 16, no. 1, pp. 103–140, 2011.

[16] C.-P. Bezemer, A. Mesbah, and A. van Deursen, "Automated security testing of web widget interactions," in *ESEC/SIGSOFT FSE*, 2009, pp. 81–90.

[17] D. Roest, A. Mesbah, and A. van Deursen, "Regression testing ajax applications: Coping with dynamism," in *ICST*. IEEE Computer Society, 2010, pp. 127–136.

[18] A. Mesbah and M. R. Prasad, "Automated cross-browser compatibility testing," in *ICSE*, 2011, pp. 561–570.

[19] A. Marchetto and P. Tonella, "Search-based testing of ajax web applications," in *Search Based Software Engineering, 2009 1st International Symposium on*. IEEE, 2009, pp. 3–12.

[20] A. Marchetto, P. Tonella, and F. Ricca, "Reajax: a reverse engineering tool for ajax web applications," *IET Software*, vol. 6, no. 1, pp. 33–49, 2012.

[21] Y.-G. Guéhéneuc, K. Mens, and R. Wuyts, "A comparative framework for design recovery tools," in *CSMR*, 2006, pp. 123–134.

[22] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege, "A systematic review of the application and empirical investigation of search-based test case generation," *IEEE Trans. Software Eng.*, vol. 36, no. 6, pp. 742–762, 2010.

[23] N. Negara and E. Stroulia, "Automated acceptance testing of javascript web applications," in *WCRE*, 2012, pp. 318–322.

[24] S. Salva and P. Laurençot, "Automatic ajax application testing," in *ICIW*, 2009, pp. 229–234.

[25] A. Mesbah, E. Bozdag, and A. van Deursen, "Crawling ajax by inferring user interface state changes," in *ICWE*, 2008, pp. 122–134.

[26] A. Mesbah and A. van Deursen, "Invariant-based automatic testing of ajax user interfaces," in *ICSE*, 2009, pp. 210–220.

[27] A. Mesbah, A. van Deursen, and D. Roest, "Invariant-based automatic testing of modern web applications," *IEEE Trans. Software Eng.*, vol. 38, no. 1, pp. 35–53, 2012.

[28] R. K. Yin, *Case study research: Design and methods*. SAGE Publications, Incorporated, 2008, vol. 5.

[29] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," in *Soviet physics doklady*, vol. 10, 1966, p. 707.

[30] S. S. Chawathe and H. Garcia-Molina, "Meaningful change detection in structured data," in *SIGMOD Conference*, 1997, pp. 26–37.

[31] L. Frantzen, J. Tretmans, and T. A. Willemse, "Test generation based on symbolic specifications," in *Formal Approaches to Software Testing*. Springer, 2005, pp. 1–15.

[32] A. M. Memon, "An event-flow model of gui-based applications for testing," *Softw. Test., Verif. Reliab.*, vol. 17, no. 3, pp. 137–157, 2007.