WEB APPLICATION TESTING: A SYSTEMATIC LITERATURE REVIEW

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

SERDAR DOĞAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2013

WEB APPLICATION TESTING: A SYSTEMATIC LITERATURE REVIEW

Submitted by **Serdar Doğan** in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems, Middle East Technical University** by,

Date: 03.09.2013

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name and Surname	**: Serdar DOĞAN**
Signature	**:**

ABSTRACT

WEB APPLICATION TESTING: A SYSTEMATIC LITERATURE REVIEW

DOĞAN, Serdar
M.S., Department of Information Systems
Supervisor: Assist. Prof. Dr. Aysu BETİN CAN
Co-Supervisor: Assoc. Prof. Dr. Vahid GAROUSI

September 2013, 88 pages

Context: The Web has had a significant impact on all aspects of our society. As our society relies more and more on the Web, the dependability of web applications has become increasingly important. To make these applications more dependable, for the past decade researchers have proposed various techniques for testing web-based software applications. Our literature search for related studies retrieved 193 papers in the area of web application testing, which have appeared between 2000 and 2013.

Objective: As this research area matures and the number of related papers increases, it is important to systematically identify, analyze, and classify the publications and provide an overview of the trends and empirical evidence in this specialized field.

Method: We systematically review the body of knowledge related to web application testing through a systematic literature review (SLR) study. This SLR is a follow-up and complimentary study to a recent systematic mapping (SM) study that has been conducted in this area. As part of this study, we pose three sets of research questions, define selection and exclusion criteria, and synthesize the empirical evidence in this area.

Results: Our pool of studies includes a set of 95 papers (from the 193 retrieved papers) published in the area of web application testing between 2000 and 2013. The data extracted during our SLR study is available through a publicly-accessible online repository. Among our results are the followings: (1) the list of test tools in this area and their capabilities, (2) the types of test models and fault models proposed in this domain, (3) the way the empirical studies in this area have been designed and reported, (4) level of rigor and industrial relevance in empirical studies and (5) the state of empirical evidence.

Conclusion: We discuss the emerging trends in web application testing, and discuss the implications for researchers and practitioners in this area. The results of our SLR can help researchers to obtain an overview of existing web application testing approaches, fault models, tools, metrics and empirical evidence, and subsequently identify areas in the field that require more attention from the research community.

Keywords: Systematic literature review, Web application, Testing

ÖZ

WEB UYGULAMASI TESTİ: BİR SİSTEMATİK LİTERATÜR İNCELEMESİ

DOĞAN, Serdar
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Yard. Doç. Dr. Aysu BETİN CAN
Ortak Tez Yöneticisi: Doç. Dr. Vahid GAROUSI

Eylül 2013, 88 sayfa

Bağlam: Web uygulamaları yaşamımızın bir çok alanını önemli ölçüde etkilemiştir. Bu uygulamalara bağımlı olduğumuz alanlar arttıkça, web uygulamalarının güvenilirliği daha önemli hale gelmiştir. Web uygulamalarını daha güvenilir kılmak için son on yılda araştırmacılar web-tabanlı yazılım uygulamalarını test etmek amacıyla çeşitli teknikler sunmuşlardır. Bu alanda yaptığımız literatür taraması sonucunda 2000 ve 2013 yılları arasında yayınlanmış 193 makale bulunmuştur.

Amaç: Web uygulaması testi alanının olgunlaşmaya devam etmesi ve bu alandaki makale sayısının giderek artması göz önünde bulundurulduğunda sistematik olarak bu alandaki yayınları belirlemek, analiz etmek, sınıflandırmak ve bu özelleşmiş alandaki çalışmaların gidişatı ile birlikte çalışmalarda ortaya konan deneysel kanıtlar hakkında genel görüntüyü ortaya çıkarmak önem kazanmıştır.

Yöntem: Bu çalışmada web uygulaması testi alanındaki bilgi birikimi "Sistematik Literatür İnceleme" (Systematic Literature Review - SLR) çalışması ile sistematik olarak gözden geçirilmiştir. Bu çalışma, bu alanda daha önce yapılmış olan sistematik eşleme (Systematic Mapping-SM) çalışmasının devamı ve tamamlayıcısı niteliğindedir. Bu çalışmanın bir parçası olarak üç araştırma soru kümesi ortaya konulmuş, çalışmaların seçim kriterleri tanımlanmış ve bu alandaki deneysel kanıtlardan sonuçlar sentezlenmiştir.

Çıktılar: Çalışma havuzumuza web uygulaması testi alanındaki 2000-2013 yılları arasında yayınlanmış 95 makale (bulunan 193 makale içerisinden elenerek) dahil edilmiştir. SLR çalışmamızda çıkardığımız veriler genel erişime açık olacak şekilde web üzerinden erişilebilir durumdadır. Bu çalışma kapsamındaki çıktılarımızın başlıcaları şöyledir: (1) bu alandaki test araçlarının listesi ve yetenekleri, (2) sunulmuş olan test ve hata modelleri, (3) bu alandaki deneysel çalışmaların nasıl tasarlandığı ve raporlandığı, (4) deneysel çalışmalardaki titizlik düzeyi ve endüstriye uygunluk durumu ve (5) bu alandaki deneysel kanıtlar.

Sonuç: Web uygulaması testi alanında ortaya çıkan eğilimler ve bu eğilimlerin bu alandaki araştırmacı ve uygulayıcılara etkileri tartışılmıştır. Elde ettiğimiz sonuçlar mevcut web uygulaması test yaklaşımları, hata modelleri, araçlar, metrikler ve deneysel kanıtlar hakkında araştırmacılara genel bir bakış sunmakla birlikte daha fazla ilgi gösterilmesi gereken alanların belirlenmesinde de araştırmacılara yardımcı olabilecektir.

Anahtar Kelimeler: Sistematik literature gözden geçirmesi, Web uygulaması, Test

To My Family

ACKNOWLEDGEMENTS

I am deeply grateful to my supervisor Assist. Prof. Dr. Aysu BETİN CAN and my co-supervisor Assoc. Prof. Dr. Vahid GAROUSI who have guided me throughout this research with their invaluable suggestions, criticisms and encouragement. They also provided whole resources, data sheets, extracted data and tools of their recent SM study which we have used to construct the basis of this study. Since this was a collaborative research they also studied with me by sacrificing their precious time without any complaint.

I am also thankful to my family for their patience during this process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASP	:	Active Server Pages
CGI	:	Computer-generated imagery
DOM	:	Document Object Model
FSM	:	Finite State Machine
HTML	:	Hypertext Markup Language
HTTP	:	Hypertext Transfer Protocol
J2EE	:	Java 2 Enterprise Edition
JSP	:	Java Server Pages
LOC	:	Lines Of Code
N/A	:	Not Applicable
PDG	:	Program Dependence Graph
PHP	:	Hypertext Preprocessor
RQ	:	Research Question
SE	:	Software Engineering
SLR	:	Systematic Literature Review
SM	:	Systematic Mapping
SUT	:	System Under Test
UML	:	Unified Modeling Language
URL	:	Uniform Resource Locator
WAT	:	Web Application Testing

CHAPTER 1

INTRODUCTION

The Web has had a significant impact on all aspects of our society, from business, education, government, entertainment sectors, industry, to our personal lives. The main advantages of adopting the Web for developing software products include (1) no installation costs, (2) automatic upgrade with new features for all users, (3) universal access from any machine connected to the Internet and (4) being independent of the operating system of clients.

On the downside, the use of server and browser technologies make web applications particularly error-prone and challenging to test, causing serious dependability threats. A 2003 study conducted by the Business Internet Group San Francisco (BIG-SF) [1] reported that approximately 70% of websites and web applications contain defects. In addition to financial costs, defects in web applications result in loss of revenue and credibility.

The difficulty in testing web applications is many-fold. First, web applications are distributed through a client/server architecture, with (asynchronous) HTTP request/response calls to synchronize the application state. Second, they are heterogeneous, i.e., web applications are developed using different programming languages, for instance, HTML, CSS, JavaScript on the client-side and PHP, Ruby, Java on the server-side. And third, web applications have a dynamic nature; in many scenarios they also possess nondeterministic characteristics.

During the past decade, researchers in increasing numbers, have proposed different techniques for analyzing and testing these dynamic, fast evolving software systems.

As the research area matures and the number of related papers increases, it is important to systematically identify, analyze and classify the state-of-the-art and provide an overview of the trends in this specialized field. In this study, we present a systematic literature review (SLR) of the Web Application Testing (WAT) research domain.

In a recent work, V. Garousi et al. conducted a systematic mapping (SM) study [2] in which 79 papers have been reviewed in the WAT domain. The current SLR is a follow-up complementary study after that SM study. This SLR continues the study started by V. Garousi et al. by focusing in depth into the empirical and evidence-base aspects of the WAT domain. This SLR study has been conducted by paying close attention to major differences between these two types of secondary studies, e.g., refer to the SLR guideline by Kitchenham and Charters [3].

The paper selection phase of this study was carried out on April and May 2013, data extraction lasted until July and data synthesis completed by the end of August 2013.

To the best of our knowledge, this study is the first SLR in the area of WAT. The remainder of this thesis is outlined as follows. A review of the related work is presented in Section 2. Section 3 explains our research methodology and research questions. Section 4 presents the results of the SLR. Section 5 discusses the main findings, trends and the validity threats. Finally, Section 6 concludes the paper mentions the future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

We classify related work into three categories: (1) secondary studies that have been reported in the broader area of software testing, (2) online repositories in software engineering, and (3) secondary studies focusing on web application testing.

2.1. Secondary Studies in Software Testing

We were able to find 24 secondary studies [2, 4-26] reported, as of this writing, in different areas of software testing. We list and categorize these studies in Table 1 along with their study areas. Based on the "year" column, we observe that more and more SMs and SLRs have recently started to appear in the area of software testing. As per our literature search, we were able to find eight SMs and five SLRs in the area, as shown in the table. The remaining 11 studies are "surveys", "taxonomies", "literature reviews", and "analysis and survey", terms used by the authors themselves to describe their secondary studies. The number of primary studies studied in each study in Table 1 varies from 6 (in [25]) to 264 (in [22]).

The recent SM study [2] reviewed 79 papers in the WAT domain and is considered the first step (phase) of the current SLR. The mapping (scoping) that has been conducted in the SM study enabled us to classify the papers and identify empirical studies. We continue in this SLR the secondary study that has been started in the SM by focusing in depth into the empirical and evidence-base aspects of the WAT domain. Note that as discussed by other researchers such as Petersen et al. [27] and Kitchenham and Charters [3], the goal and scope of SM and SLR studies are quite different and we follow those distinctions between previous SM [2] and this SLR.

Table 1- 22 Secondary Studies in Software Testing

Type of Secondary Study	**Secondary Study Area**	**Number of Primary Studies**	**Year**	**Ref.**
SM	Non-functional search-based testing	35	2008	[4]
	SOA testing	33	2011	[5]
	Testing using requirements specification	35	2011	[6]
	Product lines testing	45	2011	[7]
	Product lines testing	64	2011	[8]
	Product lines testing tools	Paper unreachable	2012	[9]
	Web application testing	79	2013	[2]
	Graphical User Interface (GUI) testing	136	2013	[26]
SLR	Search-based non-functional testing	35	2009	[10]
	Unit testing for Business Process Execution Language (BPEL)	27	2009	[11]
	Formal testing of web services	37	2010	[12]
	Search-based test-case generation	68	2010	[13]
	Regression test selection techniques	27	2010	[14]
Survey/Analysis	Object-oriented testing	140	1996	[15]
	Testing techniques experiments	36	2004	[16]
	Search-based test data generation	73	2004	[17]
	Combinatorial testing	30	2005	[18]
	SOA testing	64	2008	[19]
	Symbolic execution	70	2009	[20]
	Testing web services	86	2010	[21]
	Mutation testing	264	2011	[22]
	Product lines testing	16	2011	[23]
Taxonomy	Model-based GUI testing	33	2010	[24]
Literature review	TDD of user interfaces	6	2010	[25]

2.2. Online Paper Repositories in SE

A few recent secondary studies have reported online repositories to supplement their study with the actual data. These repositories are the by-products of SM or SLR studies and will be useful to practitioners and researchers by providing a summary of all the works in a given area. Most of these repositories are maintained and updated regularly. For instance, Harman et al. have developed and shared two online paper repositories: one in the area of mutation testing [28], and another in the area of search-based software engineering (SBSE) [29]. In three recent SM studies conducted V. Garousi et al., they also shared the paper repositories online [30-32].

We believe this is a valuable undertaking since maintaining and sharing such repositories provides many benefits to the broader community. For example, they are valuable resources for new researchers in the area, and for researchers aiming to conduct additional secondary studies. Therefore, we provide the details of our SLR study as an online paper repository [33].

2.3. Secondary Studies in Web Application Testing

We were able to find four secondary studies in the area of WAT [34-37]. A summary of these works is listed in Table 2. All four works seem to be conventional (unsystematic) surveys. Also, the size of their pool of studies is rather small, between 20 and 29 papers.

Kam and Dean [34] conducted a survey of 20 WAT papers classifying them into six groups: formal, object-oriented, statistical, UML-based, slicing, and user session-based. Alalfi et al. [35] presented a survey of 24 different modeling methods used in web verification and testing. The authors categorized, compared and analyzed the different modeling methods according to navigation, behavior, and content. Di Lucca and Fasolino [36] presented an overview of the differences between web applications and traditional software applications, and how such differences impact the testing of the former. They provide a list of relevant contributions in the area of functional web application testing. Amalfitano et al. [37] proposed a classification framework for rich internet application testing and describe a number of existing web testing tools from the literature by placing them in this framework.

All these existing studies have several shortcomings that limit their replication, generalization, and usability in structuring the research body on WAT. First, they are all conducted in an ad-hoc manner, without a systematic approach for reviewing the literature. Second, since their selection criteria are not explicitly described, reproducing the results is not possible. Third, they do not represent a broad perspective and their scopes are limited, mainly because they focus on a limited number of related papers.

Another remotely-related work is [38] in which a SLR of usability evaluation in web development has been reported, but that work does not cover the area of WAT. To the best of our knowledge, there has been no SLR so far in the field of WAT.

Table 2- Secondary studies in in web application testing

Paper title	**Ref.**	**Year**	**# of primary studies**	**Summary**
Lessons Learned from a Survey of Web Applications Testing	[34]	2009	20	A survey of 20 papers classifying them into six groups: formal, object-oriented, statistical, UML-based, slicing, and user session-based
Modelling methods for web application verification and testing: state of the art	[35]	2009	24	The study surveyed 24 different modelling methods used in web site verification and testing. Based on a short catalogue of desirable properties of web applications that require analysis, two different views of the methods were presented: a general categorization by modelling level, and a detailed comparison based on property coverage.

Testing Web-based applications: The state of the art and future trends	[36]	2006	27	Surveyed different test-specific modeling techniques, test-case design techniques, and test tools for web applications
Techniques and tools for rich internet applications testing	[37]	2010	29	A classification framework for rich internet application testing and the list of existing web testing tools

CHAPTER 3

RESEARCH METHOD

We discuss next the following aspects of our research method:

- Overview
- Goal and research questions
- Article selection
- Final pool of articles and the online repository
- Data extraction
- Data synthesis

3.1. Overview

This SLR is carried out following the guidelines and process proposed by Kitchenham and Charters [3], which has the following main steps:

- Planning the review:
    - Identification of the need for a review
    - Specifying the research questions
    - Developing a review protocol
    - Evaluating the review protocol
- Conducting the review:
    - Identification of research
    - Selection of primary studies
    - Study quality assessment
    - Data extraction and monitoring
    - Data synthesis

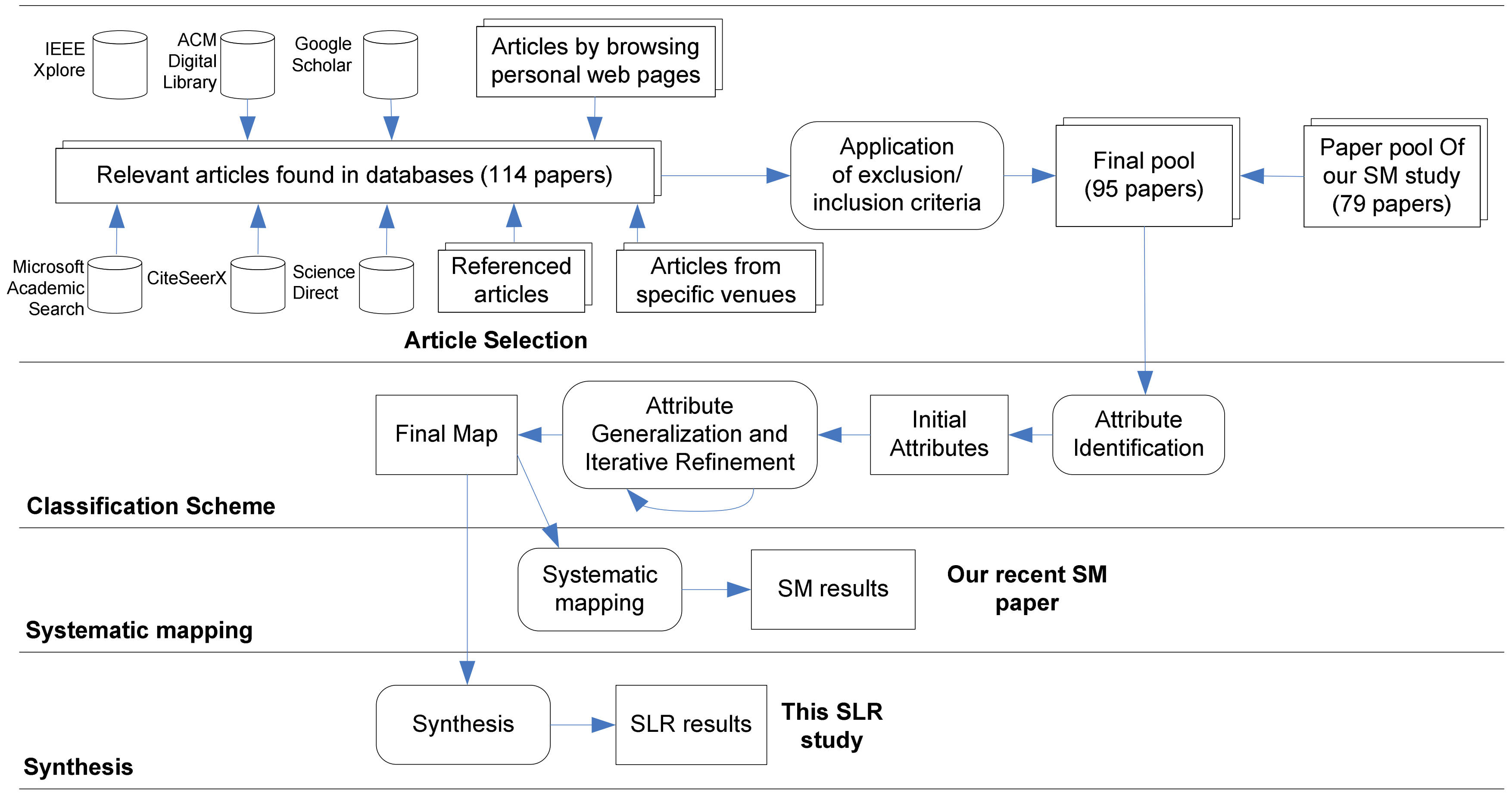


Figure 1- The review protocol used in this SLR study.

After establishing the need for the review, we specified the research questions (RQs), which are explained in Section 3.2. The process (review protocol) that we developed in the planning phase and then used to conduct this SLR study is outlined in Figure 1. The process starts with article selection (discussed in detail in Section 3.3). Then, we adapted the classification scheme of recent SM study [2] as a baseline and extended it to address the goals and RQs of this SLR. Afterwards, we conducted mapping in preparation of the SLR analysis and then synthesis to address the RQs.

We also searched the existing tools that assist the SLR process. The list of identified tools is given in Table 3. When we analyzed the web sites and existing documentation of these tools, we have seen that only "SLRGuide" and "SLuRp" cover the complete SLR process. We could not find any download information for "SPIDER". Other tools are publicly accessible online or available for download. "SLR+" and "Researchr" aim helping the primary study discovery and selection phase of SLR but do not cover the subsequent steps. Based on our investigation, we concluded that "SLRGuide" and "SLuRp" are more mature tools for assisting SLR studies than others. However in order to be able to import the data and use the existing infrastructure of recent SM study [2], we preferred to use Google Docs as our study environment which provides general purpose spreadsheets and file sharing mechanisms but highly collaborative online study environment.

Table 3- Identified SLR Tools

Tool Name	**Summary**	**SLR Support Level**	**Online**	**Public Access**
SLRGuide[1]	An open source web based SLR tool developed by Middlesex University. Supports all phases of an SLR.	Complete	Yes	Yes

[1] http://www.slrtool.org

Researchr[2]	A web site for finding, sharing, and reviewing scientific publications. Provides an online repository and enables classification according to keywords.	Partial	Yes	Yes
SLuRp[3]	A web based java application supports all phases of an SLR and it is available for download.	Complete	No	Yes
SPIDER[4]	An experimental tool targeting the medicine domain.	Partial	No	No
SLR+[5]	An open source prototype level tool for searching, downloading and grouping the studies through multiple search engines.	Partial	No	Yes

3.2. Goal and Research Questions

According to the guideline by Kitchenham and Charters [3], the goal of a SM is classification and thematic analysis of literature on a software engineering topic, while the goal of a SLR is to identify best practices with respect to specific procedures, technologies, methods or tools by aggregating information from comparative studies.

RQs of a SM are generic, i.e., related to research trends, and are of the form: which researchers, how much activity, what type of studies. On the other hand, RQs of a SLR are specific, meaning that they are related to outcomes of empirical studies. SLRs are typically of greater depth than SMs. Often, SLRs include an SM as a part of their study [27]. In other words, the results of a SM can be fed into a more rigorous

[2] http://researchr.org/
[3] https://bugcatcher.stca.herts.ac.uk/SLuRp/
[4] http://ajot.aotapress.net/content/62/3/335.full.pdf
[5] http://sourceforge.net/projects/smtp/

SLR study to support evidence-based software engineering and that is exactly what we have followed in our work.

The goal of our SLR study is to identify, analyze, and synthesize work published during the past 13 years in the field of WAT with an in-depth focus on the empirical and evidence-base aspects. Based on our research goal, we formulated three RQs. To extract detailed information, each question is further divided into a number of sub-questions, as described below:

- *RQ 1-What types of Test Models, Fault Models and Tools have been proposed?*

  It is important for new researchers to know the type and characteristics of the above artifacts to be able to start new research work in this area.

  - RQ 1.1-What types of input/inferred test models have been proposed/used?
  - RQ 1.2-What types of fault models/bug taxonomy related to web applications have been proposed?
  - RQ 1.3-What tools have been proposed and what are their capabilities?

- *RQ 2-How are the empirical studies in WAT designed and reported?*

  A study that has been properly designed and reported is easy to assess and replicate. The following sub-questions aim at characterizing some of the most important aspects of the study design and how well studies are designed and reported:

  - RQ 2.1-What are the metrics used for assessing cost and effectiveness of WAT techniques?
  - RQ 2.2-What are the threats to validity in the empirical studies?
  - RQ 2.3-What is the level of rigor and industrial relevance of the empirical studies?

- *RQ 3-What is the state of empirical evidence in WAT?*

  This RQ attempts to synthesize the results reported in the studies in order to assess how much empirical evidence we currently have. To answer this question, we address the following sub-questions:

  - RQ 3.1-Is there any evidence regarding the scalability of the WAT techniques?
  - RQ 3.2-Have different techniques been empirically compared with each other?
  - RQ 3.3-How much empirical evidence exists for each category of techniques and type of web apps?

Some of the RQs (e.g., RQs 1.1-1.3, 2.3, 3.2 and 3.3) have been raised specific to our SLR while some others have been adapted from similar SLRs in other areas of testing, e.g., RQs 2.1, 2.2 and 3.3 have been adapted from another recent SLR on search-based testing [13].

To further clarify the difference in goals and scope between the previous SM study and this SLR, the RQs of the SM study are listed in **Table 4** (adapted from [2]). We can see that the two sets of RQs are different from each other and are complimentary.

Table 4- RQs of the SM study (adapted from [2])

RQ 1 – Systematic mapping:

- RQ 1.1 – type of contribution: How many papers present test methods/techniques, test tools, test models, test metrics, or test processes?
- RQ 1.2 – type of research method: What type of research methods are used in the papers in this area?
- RQ 1.3 – type of testing activity: What type(s) of testing activities are presented in the papers?
- RQ 1.4 – test location: How many client-side versus server side testing approaches have been presented?

- RQ 1.5 – testing levels: Which test levels have received more attention (e.g., unit, integration and system testing)?
- RQ 1.6 – source of information to derive test artifacts: What sources of information are used to derive test artifacts?
- RQ 1.7 – technique to derive test artifacts: What techniques have been used to generate test artifacts?
- RQ 1.8 – type of test artifact: Which types of test artifacts (e.g., test cases, test inputs) have been generated?
- RQ 1.9 – manual versus automated testing: How many manual versus automated testing approaches have been proposed?
- RQ 1.10 – type of the evaluation method: What types of evaluation methods are used?
- RQ 1.11 – static web sites versus dynamic web applications: How many of the approaches are targeted at static web sites versus dynamic web applications?
- RQ 1.12 – synchronicity of HTTP calls: How many techniques target synchronous calls versus asynchronous Ajax calls?
- RQ 1.13 – client-tier web technologies: Which client-tier web technologies (e.g., JavaScript, DOM) have been supported more often?
- RQ 1.14 – server-tier web technologies: Which server-tier web technologies (e.g., PHP, JSP) have been supported more often?
- RQ 1.15 – tools presented in the papers: What are the names of web-testing tools proposed and described in the papers, and how many of them are freely available for download?
- RQ 1.16 – attributes of the web software under test: What types of Systems Under Test (SUT), i.e., in terms of being open-source or commercial, have been used and what are their attributes, e.g., size, metrics?

RQ 2 – Trends and demographics of the publications:

- RQ 2.1 – publication count by year: What is the annual number of publications in this field?
- RQ 2.2 – top-cited papers: Which papers have been cited the most by other papers?
- RQ 2.3 – active researchers: Who are the most active researchers in the area, measured by number of published papers?
- RQ 2.5 – top venues: Which venues (i.e., conferences, journals) are the main targets of papers in this field?

3.3. Article Selection

We followed the same article selection strategy as we had used in our SM study. We briefly discuss next the following aspects of article selection:

- Source selection and search keywords
- Application of inclusion/exclusion criteria

3.3.1. Source Selection and Search Keywords

Based on the SLR and SM guidelines [3, 27], to find relevant studies, we searched the following six major online search academic article search engines: (1) IEEE Xplore[6], (2) ACM Digital Library[7], (3) Google Scholar[8], (4) Microsoft Academic Search[9], (5) CiteSeerX[10], and (6) Science Direct[11]. These search engines have also been used in other similar studies, e.g., [7, 39].

The coverage landscape of this SLR is the area of functional testing of web applications, as well as (dynamic or static) analysis to support WAT. The set of search terms were devised in a systematic and iterative fashion, i.e., we started

[6]http://ieeexplore.ieee.org
[7]http://dl.acm.org
[8]http://scholar.google.com
[9]http://academic.research.microsoft.com
[10]http://citeseerx.ist.psu.edu
[11]http://www.sciencedirect.com

with an initial set and iteratively improved the set until no further relevant papers could be found to improve our pool of primary studies. By taking all of the above aspects into account, we formulated our search query as shown in Table 5. We searched the whole text of the studies if the search engine is capable of full text search. If the search engine not permits full text search at least title, abstract and the keywords of the studies included in the search process.

Table 5- Search Keywords

```
(web OR website OR “web application” OR Ajax OR JavaScript
OR HTML OR DOM OR PHP OR J2EE OR servlet OR JSP OR .NET OR
Ruby OR ASP OR Python OR Perl OR CGI) AND
(test OR testing OR analysis OR analyzing OR “dynamic
analysis” OR “static analysis” OR verification)
```

Since the SM study had identified 79 studies through a rigorous search process, we imported them into the SLR paper pool without an additional scanning. Note that the SM had considered the papers published until Summer 2011. We searched for more recent papers between 2011 and 2013 and included them in our candidate pool. The paper selection phase of this study was carried out on April and May 2013.

To decrease the risk of missing relevant studies, similar to previous SM studies and SLRs, we searched the following sources as well manually:

- References found in studies already in the pool
- Personal web pages of active researchers in the field of interest: We extracted the names of active researchers from the initial set of papers found in the above search engines.

All studies found in the additional venues that were not yet in the pool of selected studies but seemed to be candidates for inclusion were added to the initial pool. With the above search strings and search in specific venues, we found 114 studies which we considered as our initial pool of potentially-relevant studies (also depicted in Figure 1). At this stage, papers in the initial pool were ready for application of inclusion/exclusion criteria described next.

3.3.2. Application of Inclusion/Exclusion Criteria

Inclusion/exclusion criteria were also same as the SM study. To increase the reliability of our study and its results, the authors applied a systematic voting process among the team members in the paper selection phase for deciding whether to include or exclude any of the papers in the first version of the pool. This process was also utilized to minimize personal bias of each of the authors. The team members had conflicting opinions on four papers, which were resolved through discussions. Our voting mechanism (i.e., exclusion and inclusion criteria) was based on two questions: (1) Is the paper relevant to functional web application testing and analysis? (2) Does the paper include a relatively sound validation? These criteria were applied to all papers, including those presenting techniques, tools, or case studies/experiments.

Each author then independently answered each of the two questions for each paper. Only when a given paper received at least two positive answers (from three voting authors) for each of the two questions, it was included in the pool. Otherwise, it was excluded. We primarily voted for papers based on their title, abstract, keywords, as well as their evaluation sections. If not enough information could be inferred from the abstract, a careful review of the contents was also conducted to ensure that all the papers had a direct relevance to our focused topic.

In addition to this voting mechanism, we have checked the papers if recent version of the same study exists in our pool. If any later study found with similar authors and content, the previous one(s) excluded from the pool with consensus on exclusion decision. Also since we review primary studies, identified secondary studies moved to the related studies pool.

3.4. Final Pool of Articles and the Online Repository

After the initial search and the follow-up analysis for exclusion of unrelated and inclusion of additional studies, the pool of selected studies was finalized with 95 studies. The reader can refer to the references section at the end of this thesis for the full reference list of all primary studies. The final pool of selected studies has also been published in an online repository using the Google Docs system, and is publically accessible online at [33]. The classifications of each selected publication according to the classification scheme presented in [2] and also the empirical data extracted from the studies are also available in the online repository.

Figure 2 shows the number of papers in the pool by their year of publication. We can notice that trend has been generally increasing from 2000 until 2010, but there is a somewhat decreasing trend from 2010-2013. Note that the study was conducted in the midst of the year 2013, thus the data for this year are partial.

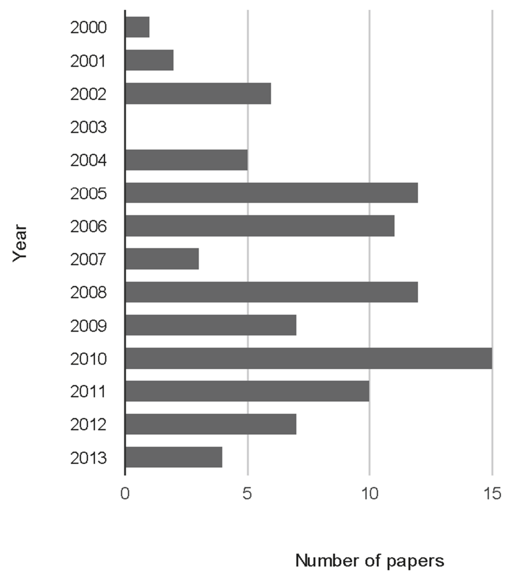


Figure 2- Annual trend of studies included in our pool

3.5. Data Extraction

As discussed above, the recent SM study of V. Garousi et al. was the first step of the current SLR. The mapping (scoping) that has been conducted in the SM study enabled us to classify the papers and identify empirical studies (using the classification "research facet" in the SM study). We conducted several new types of classifications in this SLR to further classify the primary studies as needed by several of our current RQs, e.g., RQ 1.1 (input/inferred test models), and RQ 2.1 (metrics used for assessing cost and effectiveness of WAT techniques).

The data extraction phase was conducted collaboratively among the authors and data were recorded in the online spreadsheet [33]. Data extraction phase carried out in May and June 2013.

3.6. Data Synthesis

After conducting further mapping analysis for the purpose of RQ 1 in this SLR, we conducted synthesis for answering RQs 2 and 3. To develop our method of synthesis, we carefully reviewed the research synthesis guidelines in software engineering, e.g., [40-42], and also other SLRs which had used high-quality synthesis approaches, e.g., [13, 43].

According to [40], the key objective of research synthesis is to evaluate the included studies for heterogeneity and select appropriate methods for integrating or providing interpretive explanations about them [44]. If the primary studies are similar enough with respect to interventions and quantitative outcome variables, it may be possible to synthesize them by meta-analysis, which uses statistical methods to combine effect sizes. However, in software engineering in general and in our focused WAT domain in particular, primary studies are often too heterogeneous to permit a statistical summary. Especially for qualitative and mixed method studies like ours, different methods of research synthesis, e.g., thematic analysis and narrative synthesis, are required [40].

Based on those guidelines, the fact that the primary studies in our pool were too heterogeneous and also the type of RQs in our SLR it was imperative to use thematic analysis and narrative synthesis [40-42] in this work. We followed the thematic analysis steps recommended by [41] which were as follows: extracting data, coding data, translating codes into themes, creating a model of higher-order themes, and finally assessing the trustworthiness of synthesis.

Data synthesis and reporting process completed in August 2013. During all phases of this five-month study we paid attention to refine and validate the results of previous steps.

CHAPTER 4

RESULTS

In this section, we present the results of our SLR study.

4.1. RQ 1-What types of Test Models, Fault Models and Tools have been proposed?

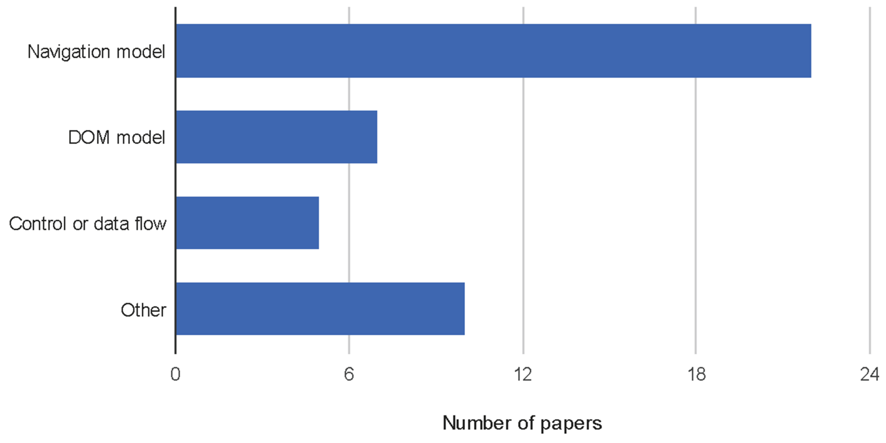
We discuss next results for RQ 1.1, 1.2 and 1.3.

4.1.1. RQ 1.1-What types of input/inferred test models have been proposed/used?

A large number of studies (40, 42%) proposed test techniques which used certain models as input or inferred (reverse engineered) them. We classified those test models as follows and the frequencies are shown in **Figure 3**:

- Navigation models (for pages): models such as finite-state machines (FSM) which specify the flow among the pages of a web application.
- Control or data flow models: these models are in unit level and are either control or data flows inside a single module/function.
- DOM models: The Document Object Model (DOM) is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents. Several approaches generated DOM models for the purpose of test-case generation or test oracles.
- Other: Any models other than the above.

The navigation models seem to be the most popular as 22 studies used a type a of navigation models. 7 studies used DOM models for the purpose of testing. Five studies used control or data flow models. The "Other" category included models such as: Program Dependence Graphs (PDGs) [S9], Database Extended Finite State Machine (DEFSM) [S22], a concept analysis model called Lattice [S25], UML models for test architecture [S26], a model called Abstract Description of Interaction (ADI) and UML models for business logic [S53], Unified Markov Models (UMMs) [S54], UML class diagram for ASP pages [S79], Request Dependence Graph (RDG) [S84] and statistical testing models [S91].



Navigation models	[S1, S7, S11, S14, S30, S32, S35, S37, S43, S51, S54, S55, S56, S58, S60, S70, S81, S82, S85, S86, S88, S93]
Control or data flow models	[S1, S31, S75, S76, S78]
DOM models	[S3, S48, S49, S59, S66, S69, S92]
Other	[S9, S22, S25, S26, S53, S54, S79, S84, S91]

Figure 3- Types and frequencies of inferred test models

Navigation models and DOM models especially used in test automation studies for testing functionality of web applications through the user interface. When we checked the testing level of studies we have seen that these navigation models are mostly used in system and integration level testing studies. On the other hand, DOM models mostly preferred in client side testing studies instead of parsing the complex HTML code. It facilitates pragmatically analyzing the content and dynamic behavior of web pages. When we look at the control or data flow models; they are mostly used in white box testing studies for representing the control flow inside a function or module.

4.1.2. RQ 1.2-What types of fault models/bug taxonomy related to web applications have been proposed?

To develop effective test techniques, it is important to understand and characterize faults specific for web applications and to develop bug taxonomies in this domain. Similar to what was conducted by another SLR in the area of testing concurrent software [45], we extracted the web fault models and bug taxonomies proposed in the primary studies, as summarized in Table 6. 21 studies discussed fault models and some of them conducted fault feeding (mutation testing) based on the proposed types of faults.

It is worth highlighting two of the studies [S36, S38] in this context. In [S36], a bug severity taxonomy was proposed which was used to assess the effectiveness and performance of a mutation testing tool for JavaScript. In [S38], a web fault taxonomy was proposed, was empirically validated, and used for fault feeding (mutation testing). In that study, 31 fault types classified under 6 categories were proposed, e.g., faults related to browser incompatibility, faults related to the needed plugins, faults in session synchronization, faults in persistence of session objects, and faults while manipulating cookies.

Table 6- Fault models/bug taxonomy related to web applications

Study	**Fault types**
S2	• Authentication • Multi-lingual • Functionality • Portability • Navigation • Asynchronous communication, • Session, • Form construction
S13	• Faults in simple link transitions • Faults in form link transitions • Faults in component expression transitions • Faults in operational transitions • Faults in redirect transitions

S16	• Data store: faults that exercise application code interacting with the data store • Logic: application code logic faults • Form: defect in form actions • Appearance: faults that change the way the page appears
S17	• Link fault: changing a hyperlink's location
S24	• Multi-tier architecture faults • GUI faults • Hyperlinked structure faults • User authentication faults
S33	• Syntactic HTML faults, e.g., element not allowed, missing attribute, end tag for unfinished element
S34	• User-event fault: Do not replay the event at the client • Message fault: Do not forward the message to the server • Timeout fault: Do not replay the timeout at the client
S36	Bug severity taxonomy: • Critical: crashes, data loss • Major: loss of functionality • Normal: some loss of functionality, regular issues • Minor: loss of functionality • Trivial: cosmetic issue
S37	Navigation faults: • Basic faults. This first category corresponds to errors that can be reproduced by simply using the application's navigation links, and possibly the web browser's back button and bookmark functionality. • Multi-window faults: Multi-window Errors. As in the previous category, these errors can be reproduced solely by using the application's and web browser's buttons; however, they require two different browser windows. • Direct URL faults. This type of errors are caused by typing a URL in the web browser's location bar from the "wrong" context.
S38	31 fault types classified under 6 categories, e.g.; • Faults related to browser incompatibility, • Faults related to the needed plugins, • Faults in session synchronization, • Faults in persistence of session objects, • Faults while manipulating cookies
S40	• Faults specific to PHP: execution failures are caused by missing an included file, wrong MySQL query and uncaught exceptions • Producing malformed HTML
S49	• DOM validity • Back-button compatibility
S50	• DOM modifications
S52	• Scripting faults: This includes faults associated with variables, such as definitions, deletions, or changes in values, and faults associated with control flow, such as addition of new blocks, redefinitions of

	execution conditions, removal of blocks, changes in execution order, and addition or removal of function calls. • Forms faults: This includes addition, deletion, or modification of a forms' name or predefined values for a name. In our target site, such faults were seeded in the sections of the scripts that dynamically generated the html code. • Database query faults: This consists of the modification of a query expression, which could affect type of operation, table to access, fields within a table, or search key or record values.
S54	• Permission denied • No such file or directory • Stale NFS file handle • Client denied by server configuration • File does not exist • Invalid method in request • Invalid URL in request connection
S57	• Blank page • 404 error • Cosmetic • Language error • CSS error • Code on the Screen • Wrong page / no redirect • Authentication • Permission • Session • Search • Database • Failed upload • Missing image
S59	• Dead or unreachable JSP code, which often indicates unintended behavior • Calls to built-in functions with a wrong number of arguments or with arguments of unexpected types • Uses of the special JavaScript value undefined (which appears when attempting to read a missing object property) at dereferences or at function calls
S63	• Faults in PHP programs: execution faults, • HTML faults: these involve situations in which generated HTML code is not syntactically correct, causing them to be rendered incorrectly in certain browsers.
S69	Ajax faults: • Incorrect manipulation of the DOM, for example deriving from assumptions about the DOM structure which become invalid during the execution because of page manipulation by JavaScript code. • Inconsistency between code and DOM, which makes the code reference an incorrect or nonexistent part of the DOM. • Unintended interleaving of server messages

	• Swapped callbacks • Executions occurring under incorrect DOM state
S84	• GUI faults • Database operation faults • Navigation faults
S92	• Layout issues • Differences in element position • Size • Visibility or appearance • Functionality issues

4.1.3. RQ 1.3-What tools have been proposed and what are their capabilities?

52 of the 95 papers (54%) presented (mostly prototype-level) tool support for the proposed WAT approaches. In order to count a study as proposing a tool, we looked if the tool is proposed in that study. We did not count the tools used but not proposed in the study itself. Also the used existing commercial, open source or academic tools are not in the scope of this research question. We thought that a natural question to ask in this context is whether the presented tools are available for download, so that other researchers and practitioners could use them as well. We only counted a presented tool available for download if it was explicitly mentioned in the paper. If the authors had not explicitly mentioned that the presented tool is available for download, we did not conduct internet searches for the tool names. Only 11 of the 52 presented tools (21%) were available for download. We also wanted to know whether more recently-presented tools were more likely to be available for download. To assess this, Figure 4 shows the annual trend of the number of tools presented in the papers and whether they are available for download. We can notice that, in the papers presented after 2008, more and more tools are available for download, which is a good sign for the community.

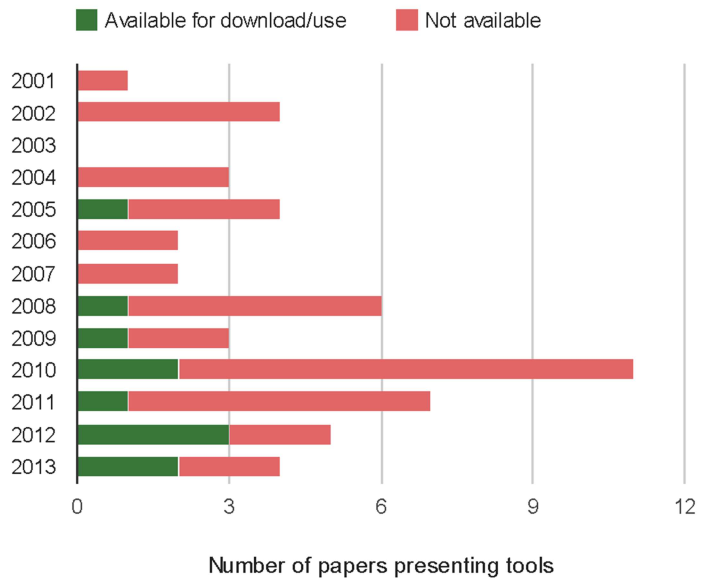


Figure 4- Number of tools presented in the papers over the years

To further answer RQ1.3, we extracted the features and capabilities of the tools available for download. They are reported in Table 7.

Table 7- Features of the tools available for download/use

Tool name	**Study**	**Features**	**URL**
MBT4Web	S7	Model-based testing framework for web applications	www.mbt4web.fh-stralsund.de
MUTANDIS	S36	JavaScript mutation testing tool that leverages static and dynamic program analysis to guide the mutation generation process towards parts of the code that are error-prone or likely to influence the program's output	www.github.com/saltlab/mutandis
ATUSA	S48, S66	Dependent on Crawljax. Automatically testing UI states of AJAX	www.crawljax.com
Crawljax	S66	Crawling Ajax-based web applications and reverse engineering of their FSMs	www.crawljax.com
JSART	S50	Regression testing of JavaScript code based on assertions	www.salt.ece.ubc.ca/content/jsart
Tool-suite: (CreRIA, CrawlRIA, Test Case	S67	A tool-suite for dynamic analysis and automatic regression testing of JSP applications	wpage.unina.it/ptramont/downloads.htm

Generator, Test Case Reducer, DynaRIA)			
reAJAX	S85	A tool for extracting the FSM of Ajax applications through dynamic and static code analysis	selab.fbk.eu/marchetto/tools/ajax/testing1/experimentData.zip
WebMate	S93	A tool-set which systematically explores all interactions in a web application and devises a usage model with all distinct behaviors of the application. The tool also creates tests that cover all distinct behaviors in the usage model to provide fully automatic cross-browser compatibility testing.	www.degesso.de
WebVizOr	S94	A visualization tool for applying automated oracles and analyzing test results of web applications	www.eecis.udel.edu/~hiper/webvizor
Web Portal In-container Testing (WIT)	S95	A tool for in-container testing of web portals. Using the aspect technology, the test code is injected into the application code allowing the tests to run in the same environment as the portal application.	sourceforge.net/projects/wit-ict/files/wit-ict

4.2. RQ 2-How are the empirical studies in WAT designed and reported?

This research question aims to investigate and assess the design and reporting of empirical studies in the domain of WAT. To answer this question, we further divided it into three sub-questions. By answering each sub-question individually, we will answer the main research question. Though the results are presented in tables that summarize the main findings, the reader can obtain a breakdown of which papers led to these findings in the online paper repository [33].

4.2.1. RQ 2.1-What are the metrics used for assessing cost and effectiveness of WAT techniques?

Assessing the cost-effectiveness of WAT techniques is an important objective of empirical studies in this area. We discuss next the list of metrics used for assessing cost and effectiveness.

4.2.1.1. Cost Metrics

Based on the type of metrics used in the primary studies, we classified cost metrics into four categories: (1) effort/test time, (2) test-suite size, (3) memory space, and (4) other. Frequencies of the cost metrics used in the empirical studies are shown in Table 8 and discuss them next.

Table 8- Frequencies of cost measures across empirical studies

	Effort / test time	**Test-suite size**	**Memory space**	**Other**	**None**
Empirical studies (N=58)	26	17	7	3	22
	44%	29%	12%	6%	37%

Effort / test time

26 studies measured test effort/time and, by doing so, most of them aimed at assessing the scalability and practicality of the approaches. For example, [S2] measured "preparation" time of the approach it proposed which was the time required (in man-hours) to prepare the testing environment (e.g., model extraction, requirements analysis, model construction, probes insertion, etc.).

In [S5], a module of the test-case generation tool was a constraint solver and a part of the empirical study was to measure the running time (in seconds) of the solver to ensure that it will scale up to large SUTs. [S10] is an empirical comparison of three test-suite reduction techniques for user-session-based testing of web applications: concept analysis, and two requirements-based approaches. Their study carefully measures the execution time of each of the phases and compares the time performance of the three techniques. [S14] measured and reported the crawling time (in minutes) needed for the tool to run. [S16] proposed automated oracles for web applications and compared their execution times on four case study SUTs. In the case-study of [S16], the slowest oracle executed in 14 minutes on average, whereas replaying the test suite took about 90 minutes.

[S48] measured both the amount of manual effort by human in utilizing the proposed approach and also the tool's performance. Some studies measured the amount of overhead time needed to run certain tool, e.g., [S50] presented a tool called JSART for JavaScript assertion-based regression testing and measured the extra time needed to execute the application while assertion checks are in place. [S92] also used execution time to assess scalability. Nine web pages were evaluated and it was reported that analysis of each of them took the proposed test tool less than five minutes to complete.

<u>Test-suite size</u>

Test-suite size is one of the oldest and most conventional cost metrics in software testing, which was measured in 17 studies. In most of the cases, the goal of measuring test-suite size was to correlate it with effectiveness metrics, e.g., coverage and mutation score. For example, in [S3], one of the RQs was to find out the level of code coverage achieved by each of the test generation algorithms under consideration that each is allowed to generate the same number of tests.

Another group of works measuring test-suite size were those which aimed at the classical problem of test suite reduction, i.e., to reduce the test-suite size while keeping the same fault detection effectiveness. Six studies [S10, S12, S47, S67, S69, S89] had the above-mentioned objective. For instance, [S10] is an empirical comparison of three test-suite reduction techniques for user-session-based testing. [S12] examined the impact of three test-suite reduction techniques on cost/benefit of test suites.

<u>Memory space</u>

For test techniques to be practical, they should have reasonable memory space requirements and should scale up for large SUTs. Seven studies [S10, S16, S17, S18, S28, S51, S90] measured this metric. We discuss [S16, S17] as two representative studies.

[S16] proposed automated oracles for web apps and compared their memory space requirements on four SUTs. [S17] proposes an automated framework for user-session-based testing of web-based software that focuses on scalability and evolving the test suite automatically as the application's operational profile changes. To quantitatively measure scalability, the study reported the memory costs of replaying the test suite for the two case study systems (consuming 4.2 and 18 MB of RAM).

<u>Other</u>

Three studies [S2, S4, S68] used/proposed other types of cost metrics, which were related to cost and complexity. [S2] measured test-suite complexity which was defined as the number of steps (test commands) required to execute whole suite. Note that this metric is different than test-suite size. [S4] measured number of states and number of edges in the navigation model, a specific test model that was generated for the purpose of testing. [S68] reported size metrics of the test models, finite-state machines, built in the case study, which included the number of states and edges.

4.2.1.2. Effectiveness Metrics

Distribution of effectiveness measures across empirical studies is shown in Table 9. Code coverage was the most popular metric (used in 28 studies) followed by detection of injected faults (mutation score) which is used in 27 studies. We classified coverage into three categories: coverage of code, coverage of models (e.g., FSM), and coverage of other artifacts (e.g., URLs). Each metric type is discussed next.

Table 9- Frequencies of effectiveness measures across Empirical Studies

	Empirical studies (N=58)	
Coverage (code)	28	29%

Coverage (model-based requirements)	9	9%
Coverage (other)	15	16%
Detecting real faults	17	18%
Detecting injected faults	27	28%
Other	10	11%
None	9	15%

Code coverage metrics seem to be quite popular in assessing effectiveness of WAT techniques. 28 papers used at least one type of code coverage in their evaluations. We further counted the number of papers using each type of code coverage and results are shown in Table 10. Statement (also called line or node) coverage was the most widely used coverage metric. Both data-flow and control-flow metrics have been used in the studies.

Table 10- Frequency of code coverage metrics used for the purpose of effectiveness measurement

Metric	**# of studies**	**% of empirical studies**
Statement/line/node	20	34%
Block	3	5%
Branch	6	10%
Condition	1	2%
Path	3	5%
Functions	2	3%
Method	1	2%
All uses	1	2%
All defs	1	2%
All def-use	2	3%

We distinguished model-based coverage metrics from code-based ones since they were based on inferred models of web applications, e.g., FSM models. We identified four types of metrics in this category: state coverage, model edge (transition) coverage, model path coverage and prime path coverage (defined in [S62]). Table 11 shows the references.

Table 11- Model coverage metrics used for the purpose of effectiveness measurement

Model coverage metrics	**Studies**
State coverage	S43, S49, S67, S69
Edge (transition) coverage	S4, S43, S67, S69, S85
Path coverage	S43
Prime path coverage	S62

15 empirical studies presented and utilized coverage metrics which were not code or model-based. We categorized them under the "Coverage (other)" category in in Table 9. We have summarized the list and brief definition of those metrics in Table 12.

Table 12- Other coverage metrics

Coverage metrics	**Studies**
• Use-case coverage: number of use cases exercised by a test suite.	S2
• Event space coverage: number of events in the GUI space of the SUT exercised by a test suite.	S5
• All-URL coverage: covering each URL of the application at least once.	S10
• Page coverage: every page in the SUT is visited at least once in some test case. • Hyperlink coverage: every hyperlink from every page in the site is traversed at least once.	S11, S77
Three database-specific coverage criteria: • Page access coverage: measures the adequacy of test cases for ensuring that all server pages are executed at least once. • SQL statement coverage: measures the adequacy of test cases to insure that all possible SQL statements, including dynamically constructed ones, are tested at least once. • Server environment variable coverage: Server environment variables are variables returned by HTTP forms on generated pages using GET or POST. Server environment variable coverage measures the adequacy of test cases to insure the coverage of all server environment variables at the level of the web application.	S20

• Single URLs: requires all URLs in the base URL set of a SUT to be covered at least once by the test suite. • URL seq2: requires every transition from each URL to any other URL in the SUT to be covered at least once by the test suite. • URL names: requires all possible URLs together with variable names (e.g., login.asp? email&password) to be covered at least once by the test suite. • URL seq2 names: requires covering "URL seq2" criterion and also all the possible variable names on each pair of URLs. • URL names values: A test set should capture names and values of variables responsible for the dynamic behavior of a web application (e.g., login.asp? email="test@gmail.com"& password="pass"). • URL seq2 names values: A test set satisfying URL seq2 names values should capture control flow as well as the names and values of variables responsible for changing URL control flow.	S28
• All rules coverage: Given a set of formal business rules, a test suite should cover them all. An example of a formal business rule: card_type_record_page.name≠EMPTY ∧ explored(card_type_record_page.insert)	S43
• Input-parameter coverage: covering all possibilities of input parameters using black-box approaches, e.g., equivalence classing	S45
• Command-form coverage: A coverage criterion specific for database applications that focuses on adequately exercising the interactions between an application and its underlying database. It has been defined in [55].	S45, S64
• Template variable coverage: Covering all the template variable in HTML pages.	S62
• Input validation coverage (IVC): At least one path in the program's CFG w.r.t. the validation of inputs has been covered.	S76
• All-hyperlinks • All-input-GUI • All-events	S78
• Pages • Index pages, • Web object with high coupling	S81
• Request coverage • Data dependence transition relation coverage • Link dependence transition relation coverage	S84

10 studies used metrics other than coverage or mutation score for the purpose of effectiveness measurement. Table 13 lists those studies and the names of the metrics. Each metric is briefly discussed next.

Table 13- Other metrics used/proposed for the purpose of effectiveness measurement

Metrics	**Studies**
False positives, False negatives	S30, S21, S50 , S57, S66
Precision, Recall	S21, S50 , S57, S64
Number of interfaces discovered	S45
Number of DOM violations	S49
Reliability growth	S54
Test path reduction rates	S9

As shown in Table 13, five studies measured the accuracy of their proposed approaches by measuring false positives and/or false negative metrics. A false positive is a mistakenly reported fault. A false negative is an undetected real fault. Three of those studies have gone further and measured the precision and recall, based on false positive and false negative metrics.

[S45] presented an approach for test-case generation for web applications using automated interface discovery. To assess effectiveness, besides a few coverage metrics, the study measured the number of interfaces discovered in its case study. [S49] presented an invariant-based automatic testing approach for AJAX applications. A metric used for assessing the effectiveness of the approach was the number of DOM violations, given a set of invariants.

In [S54], usage and reliability of web applications was measured and modeled for the purpose of statistical web testing. Test effectiveness was measured by the reliability change (or growth) through the testing process. This reliability change was evaluated by software reliability growth models (SRGMs), i.e., Goel-Okumoto model.

[S9] presented a regression testing approach for PHP web applications. As we know, test reduction is an important goal in regression testing. Thus, this study measured a relevant metric, i.e., test path reduction rates.

Detecting injected faults metric is the second mostly used effectiveness metric in 28% of the empirical studies. In order the compare the fault detection capability of techniques, tools etc. source of the subject application is modified for injecting artificial faults which is called mutation. Then the number, severity or other attributes of detected injected faults compared for evaluating effectiveness. E.g., [S2] evaluates the fault detection capability of their state-based testing technique, developed to test AJAX-based applications, by comparing the revealed injected faults with existing other web testing techniques. Another study [S16] presents a suite of automated oracle comparators for testing web applications and evaluates the effectiveness of these comparators by comparing the number of injected faults each oracle correctly identifies.

17 studies used detecting real faults metric for effectiveness evaluation. This metric is similar to detecting injected faults metric but here the faults are not artificial. These studies use real subject applications without any mutation and evaluate the capability of an approach or tool of revealing already existing real faults in the application. [S3] proposes a framework for automated testing of JavaScript web applications using feedback-directed automated test generation for JavaScript. They experimented their framework on 10 open source web applications and evaluated the effectiveness by using achieved code coverage as well as the number of detected real HTML validity and runtime errors for their different algorithms.

4.2.1.3. Usage of Multiple Metrics

We noticed that many studies have measured and reported multiple metrics. Furthermore, we hypothesized that studies with more mature research facet (validation research vs. more mature evaluation research approaches) might use more metrics in their evaluations. To evaluate this hypothesis, we plotted the number of cost and effectiveness metrics in each study, grouped by the two above research facet types as shown as an individual-value plot in **Figure 5**.

Although we do not notice statistically significant differences between the research facet types, it is interesting to observe that, generally, for the number of effectiveness metrics, at least, studies with evaluation research approaches tend to use slightly more metrics compared to validation research studies.

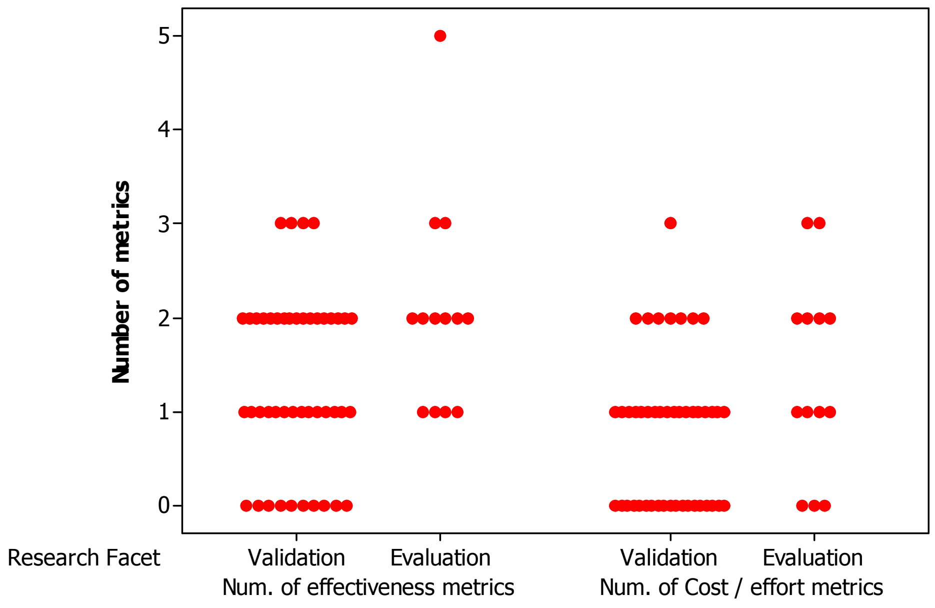


Figure 5- Usage of multiple metrics in each of the studies, as an individual-value plot

4.2.1.4. Pairs of Cost and Effectiveness Metrics

In a follow-up to usage of multiple metrics in each study, we wanted to find out which pairs of cost and effectiveness metrics have been used more often in the studies. To analyze this objective, we counted the number of studies which had used each pair of the metrics discussed in previous sections. The results are shown using a bubble chart in Figure 6.

As we can observe, the three most widely used metric pairs are:

- (detecting injected faults, effort)
- (detecting injected faults, test-suite size)
- (code coverage, effort)

These pairs of metrics are commonly used in the general software testing literature for the purpose of assessing test effectiveness, e.g., [46, 47] and it thus seems that researchers in the WAT domain have adapted those metrics as well.

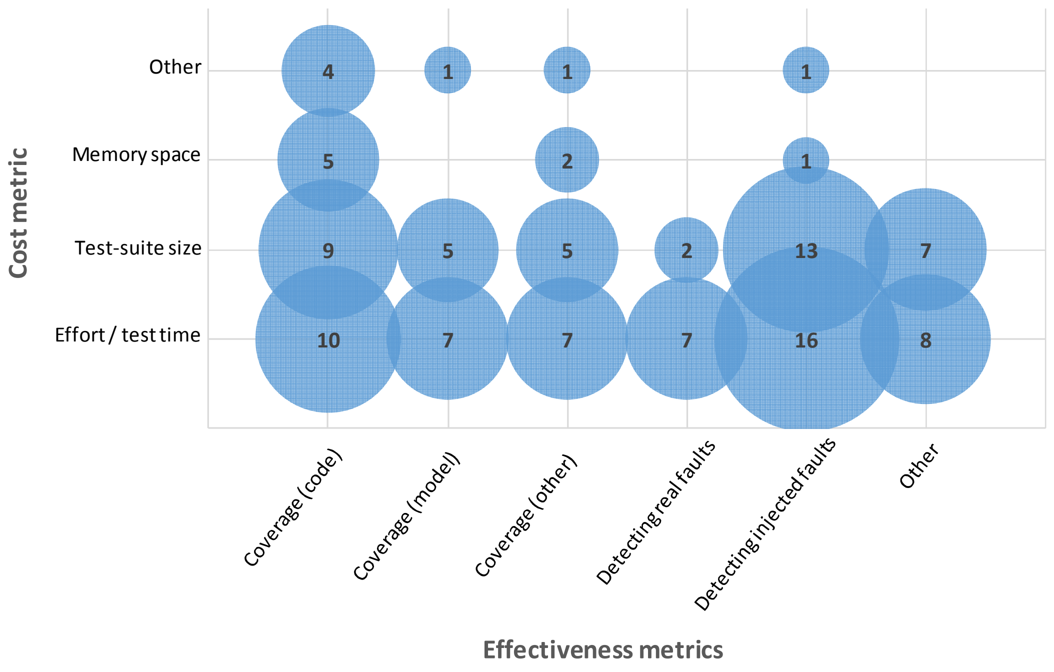


Figure 6- Bubble-chart of pair of cost and effectiveness metrics used in the studies

4.2.2. RQ 2.2-What are the threats to validity in the empirical studies?

Identification and discussion of validity threats is the one of the most important aspects of empirical research in software engineering [48]. Several studies have presented classifications and have analyzed the validity threats in software engineering in order to guide researchers on how validity threats are analyzed and alleviated in empirical software engineering [49, 50].

In our set of primary studies, most of the empirical studies mentioned these threats explicitly, while some have referred to these threats as limitations [S33, S37, S62, S91, S92, S93]. We have extracted the threats in instances when authors explicitly identified them as validity threats. The types of validity threats are classified into the following four types [48]:

- Internal validity threats: Threats which may have affected the results and have not been properly taken into account.
- Construct validity threats: Threats about the relationship between theory and observation(s).
- Conclusion validity threats: Possibility to derive inaccurate conclusions from the observations.
- External validity threats: Threats that affect the generalization of results.

We extracted the threats from the papers which we have identified as validation and evaluation research. In those papers that have a "Threats to Validity" section but have not mentioned the type of threats explicitly, we have identified the threat types according the classification above. In order to synthesize the validity threat data, we needed a threat classification in higher level of granularity. To the best of our knowledge, there is no such classification for validity threats of WAT studies in the literature. To overcome this problem, we classified the threats according to their reasons by defining short phrases for each type of threat based on the description in the paper text. E.g. "representativeness of SUTs", "error in human-based identification". This brought us a new classification scheme for validity threat types as given in Table 15-Table 18.

Since RQ 2.2 focuses on empirical studies, the papers which are categorized as validation or evaluation research were taken into account when analyzing the validity threats. Out of the 13 studies categorized under "Evaluation Research", all of them identified at least one threat, while out of the 45 studies categorized under "Validation Research", only 23 (51%) of them identified at least one threat (Table 14).

Table 14- Validity threats in Empirical Studies

Type of Paper Research Facet	**Number of Empirical Studies**	**Identified at Least one Validity Threat**	**Percent**
Evaluation Research	13	13	100%
Validation Research	45	23	51%
Total	58	36	62%

During the extraction of validity threats, we realized that some threats were categorized under more than one category, e.g., "representativeness of seeded faults" was referred to as an external threat in three studies [S17, S28, S65], but most of the other studies categorized this threat as an internal threat. According to our understanding and interpretation of the empirical software engineering literature, we treated this threat as an internal threat.

As Figure 7 shows, external and internal validity threats are the most addressed threats. Construct and conclusion threats were identified in only 9 and 6 studies respectively.

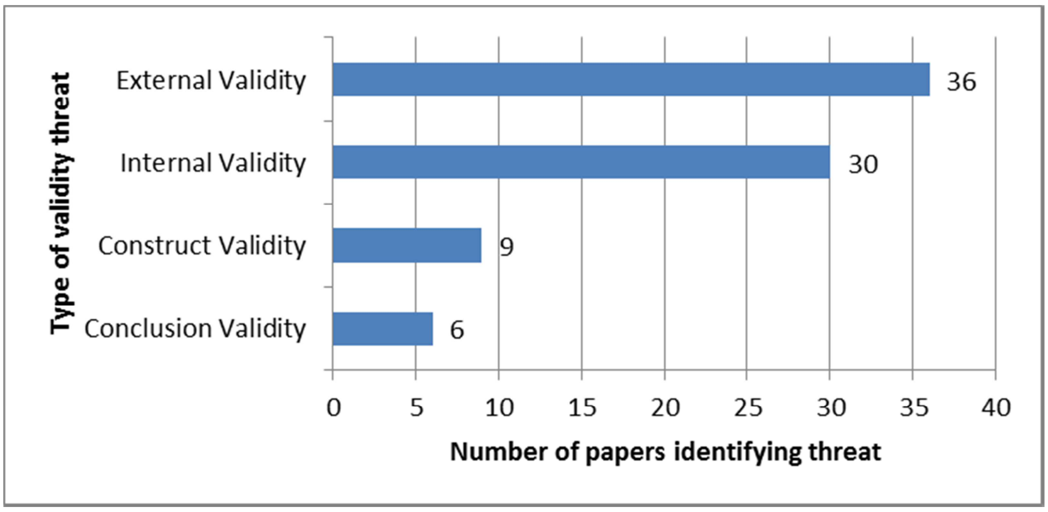


Figure 7- Number of papers identifying threats in each type

We also wanted to assess the number of identified threat types in each single study. Figure 8 shows the histogram of the data. 22 of the 58 empirical studies did not identify any type of validity threats, while 5 studies identified one type of threat and 3 studies [S2, S29, S38] identified all four types.

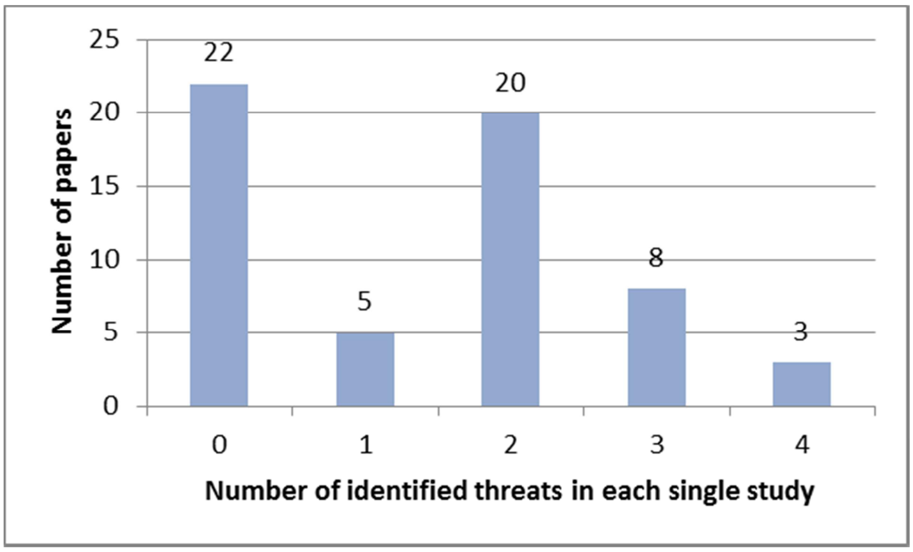


Figure 8- Number of identified threats in each single study

We list in Table 15 the list of internal validity threats identified in the empirical studies. In total, 20 types of internal validity threats were identified in 30 studies. "Representativeness of injected faults/mutations" appeared in 17 studies and is the most addressed internal validity threat. "Dependence on third party tools/libraries" and "Error in human-based identification" are the next two mostly addressed threats.

Table 15- Internal validity threats identified in the studies

Type	Explanation	Studies	Count
Representativeness of injected faults/mutations	• Selection process for faults may be affected by researchers' bias (e.g. from fault models they have) • Hand-seeded faults • Type of mutations • Faults may not represent real world situations • Even distribution of mutations	S2, S13, S16, S17, S21, S28, S34, S40, S47, S48, S49, S52, S57, S63, S65, S85, S90	17

Dependence on 3rd party tools/libraries	• Faults in used 3rd party tools/libraries • Change in implementation of 3rd party tools/libraries may affect results	S14, S45, S48, S61	4
Error in human-based identification	• Some steps and identifications done by human • Human training bias also included in this category	S21, S29, S36, S57	4
Simplicity of the SUTs	• More complex SUTs may change the empirical results	S10, S17	2
Subjectivity in applying techniques	• Proposed technique includes human dependent manual steps	S2	1
Not considering all important metrics	• e.g. cost effectiveness	S52	1
Representativeness of human subjects	• The experience level and expertise of human subjects in manual steps of the study	S57	1
Faults in implemented tool	• Quality of the proposed tool	S61	1
Different behavior of browsers	• Study applied on one or limited number of browsers, results may change on other browsers.	S16	1
Limited source of information to derive test artifacts	• e.g. lack of a large number of user sessions that constitute the requirements universe.	S28	1
Small number of classifiers	• In taxonomy studies, having more human classifiers would lead to better results	S38	1
Automatically generated inputs	• Automatically generated or classified information to generate test artifacts	S51	1

Time costs depending on case	• Execution time depends on the machine used or test case characteristics	S65	1
Error in oracle comparators	• Oracle comparators can have false positives and false negatives	S65	1
Test suite size	• Size of test suite affecting the results	S85	1
Biased SUT selection	• Selection process for the subjects is not randomized. Researcher's bias affects the choice.	S93	1
Selection of criteria to generate test artifacts	• e.g. results are influenced by the used criterion to extract test cases from the built models	S70	1
Initial results affecting subsequent ones	• e.g. failure in a test case affects next one	S66	1
Dependence to database state and configuration	• Behavior of SUTs which are using a database may change according to different state and configuration.	S19	1
Error in human based code modification	• Manually changing statements in a program during a phase of the approach, • Removing or fixing any dynamic environment variables manually.	S9	1
Total number of papers identifying an internal validity threat		30	

External validity threats are one of the mostly addressed types of threats. We list in Table 16 the list of external validity threats identified in the empirical studies. 7 distinct types of external validity threats were identified in 36 papers. According to the results, "Representativeness of SUTs" is the mostly addressed external validity threat with 55% of the empirical studies in our pool.

Table 16- External validity threats identified in the studies

Type	Explanation	Study	Count
Representativeness of SUTs	• Size, technology, context, type etc. of subject applications cannot be generalized to whole target domain.	S2, S3, S9, S12, S13, S14, S16, S17, S18, S19, S21, S28, S29, S34, S47, S36, S45, S47, S48, S51, S52, S57, S59, S61, S63, S65, S66, S69, S70, S85, S90, S93	32
Need for more real-world studies	• Empirical comparison needs to be evaluated further by experimenting with more real-world web applications	S4, S10, S49	3
Lack of comparison to other studies	• Not comparing the study with related studies.	S36, S40	2
Scalability not sure	• Empirical study not applied on realistic large scale subjects	S2	1
Representativeness of bug dataset	• (Mostly in taxonomy studies) The input data set is relatively small.	S38	1
Interaction style of web app users	• Participants' interaction patterns with the system are not sufficient for representing potential users and real world system characteristics.	S52	1
Total number of papers identifying an external validity threat		36	

We list in Table 17 the list of construct validity threats identified in the empirical studies. 10 distinct types of construct validity threats were identified in 9 papers.

Table 17- Construct validity threats identified in the studies

Type	Explanation	Study	Count
Minimization of false positives at the risk of not detecting faults.	• Trying to minimize false positives rises the number of false negatives	S90	1
Inadequacy of test input generation strategy	• Efficiency and effectiveness of the approach used in generation of test inputs and artifacts has important impacts on the results.	S45	1
Subjectivity in bug classification	• Human based classification of faults	S38	1
Inadequate bug location report	• Fault localization is weak	S40	1
Not considering the severity of the faults	• E.g. potential impact of fault severity on the used techniques and oracle comparators.	S17	1
Subjectivity in choosing metrics	• Human based metric selection may include a level of subjectivity	S2	1
Faults in implemented tool	• Depends on the quality and the reliability of implemented tool	S13	1
Dependence on crawler's capabilities,	• Quality of 3rd party crawler tool affects would affect the results.	S29	1
Dependence on definition of metrics	• E.g. different definitions of crawlability metrics may lead to different results	S29	1
Dependence on third party tool	• E.g. dependence to a third party tool for measuring coverage	S93	1
Total number of papers identifying a construct validity threat		9	

The results of conclusion validity threats are similar to construct validity threats. There is no trend that shows a type of threat is addressed more than others. Also this kind of validity threats are the least addressed threats in the studies. We list in Table 18 the list of conclusion validity threats identified in the empirical studies. 7 distinct types of conclusion validity threats were identified in 6 papers.

Table 18- Conclusion validity threats identified in the studies

Type	Explanation	Study	Count
Not considering all types of effort spent	• E.g. experiment does not consider the effort required to select inputs and oracles of each test case but only the number of the test cases.	S70	1
Results rely on interpretation of metrics	• E.g. taxonomy assessed through metrics manually.	S38	1
Results rely on human based classification	• E.g. human based classification of faults in taxonomy studies.	S38	1
Not using statistical tests	• E.g. To reject the null hypotheses	S2	1
Not including fault severity	• conclusions of the experiment could be different if the results were weighted by fault severity	S12	1
Dependence on used statistical technique	• Using different statistical techniques may affect the outcome	S29	1
Need to maintain the state of the application	• Applying same methods and techniques on different state of same SUT may provide different results	S10	1
Total number of papers identifying a conclusion validity threat		6	

We also asked if there is any trend between year of publication and the discussion of validity threats. Results are given in Figure 9. We could not see any visible trend between threat identification and publications years of papers.

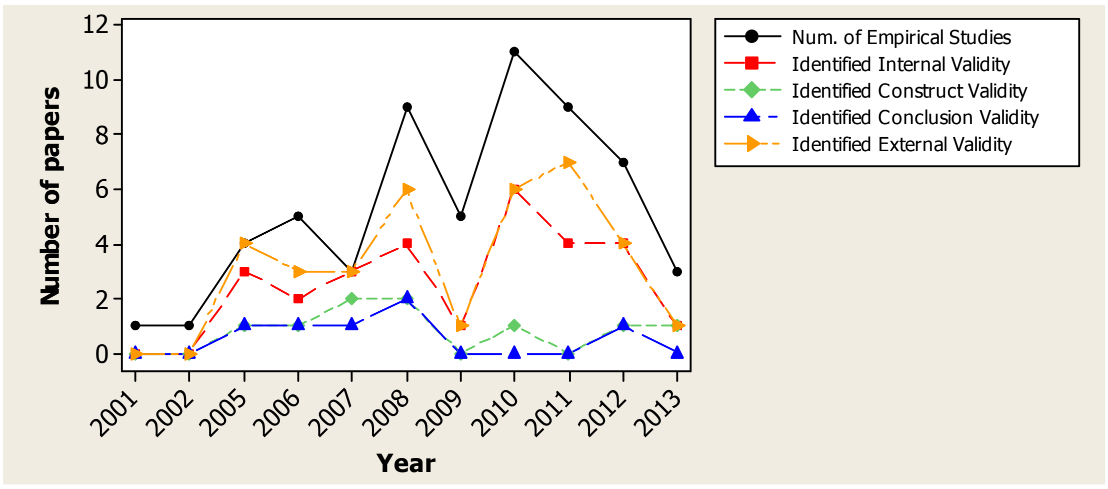


Figure 9- Trend of the number of empirical studies and type of identified threats over the years

When we consider all types of threats, total number representativeness of SUTs and representativeness of injected faults/mutations is almost equal to the total number of whole others. From this point, it is easy to say that these two are the main threats to validity of WAT studies.

4.2.3. RQ 2.3-What is the level of rigor and industrial relevance of the empirical studies?

Ivarsson and Gorschek presented in [51] a method for evaluating rigor and industrial relevance of empirical studies. The authors argue that, to impact industry, software engineering researchers developing technologies in academia need to provide tangible evidence of the advantages of using them. This can be done trough step-wise validation, enabling researchers to gradually test and evaluate technologies to finally try them in real settings with real users and applications. The evidence obtained, together with detailed information on how the validation was conducted, offers rich decision support material for industry practitioners seeking to adopt new technologies and researchers looking for an empirical basis on which to build new or refined technologies.

The model presented in [51] approached the measurement of rigor and industrial relevance as follows. For assessing rigor of an empirical study, three aspects should be measured: (1) context described, (2) study-design described, and (3) validity discussed. The rubric used for these measurements is: strong description (1), medium description (0.5), and weak description (0).

For the industrial relevance of an evaluation, the model proposes two aspects. First, the realism of the environment in which the results are obtained influence the relevance of the evaluation. Three aspects of evaluations are considered in evaluating the realism of evaluations: subjects, scale and context. Second, the research method used to produce the results influence the relevance of the evaluation. A diverse set of research methods is included in the model to cover a wide range of activities from application (test/illustration) of a technology to experiments and any sort of empirical evaluation.

The scoring rubrics used to assess these relevance aspects as presented in [51] are shown in Table 19. We made a slight adjustment to the "scale" aspect: if the sum of LOC of the SUTs in a study was less than 1,000 LOC, we assigned 0, if it was between 1,000 and 10,000 LOC, we assigned 0.5, and if larger than 10,000 LOC, we assigned 1. Note that we assessed rigor and industrial relevance for the 58 empirical studies only.

Table 19- Scoring rubric for evaluating relevance (adapted from [51])

Aspect	Contribute to relevance (1)	Do not contribute to relevance (0)
Context	The evaluation is performed in a setting representative of the intended usage setting, i.e., industrial setting.	The evaluation is performed in a laboratory situation or other setting not representative of a real usage situation.
Users/ subjects	The subjects used in the evaluation are representative of the intended users of the technology, i.e., industry professionals.	The subjects used in the evaluation are not representative of the envisioned users of the technology (practitioners).

Scale	The scale of the applications used in the evaluation is of realistic size, i.e., the applications are of industrial scale.	The evaluation is performed using applications of unrealistic size.
Research method	The research method mentioned to be used in the evaluation is one that facilitates investigating real situations and that is relevant for practitioners, e.g., action research	The research method mentioned to be used in the evaluation does not lend itself to investigate real situations, e.g., conceptual analysis laboratory experiment

The histograms of Figure 10 show the data. Note that, for the case of "Users/subjects", the chosen rubric value was N/A (not applicable) since many studies in our pool did not involve human-based experiments. We discuss below our main observations based on the two histograms.

In terms of rigor, we can observe that:

- In most of the empirical studies (47 of the 58), the context of the empirical study has described to a degree where a reader can understand and compare it to another context.
- Study design has been also explained well in most of the studies (43 of the 58).
- In terms of validity discussion, 17 of the 58 studies have not discussed the validity of the study at all, 19 have explained very briefly including 5 papers which have mentioned the limitations [S33, S37, S62, S91, S92, S93], and 22 have explained in enough detail.

For industrial relevance of studies, we can observe that:

- In terms of context, 40 of the 58 studies were performed in a laboratory setting, while 18 were conducted in industrial context or on an industrial real web application

- As long as the research methods are concerned, none of the studies used methods relevant for practitioners, e.g., action research.
- Users/subjects: For 55 papers, the aspect of users was N/A. For the remaining three studies [S18, S38, S91], industry users were involved. In [S18], a group of 14 participants exercised the SUTs and session data were then collected. In [S38], expert testers were hired to classify defects. In [S91], industry professionals helped in gathering data.
- In terms of the scale aspect, 35 of the 58 studies used SUTs larger than 10,000 LOC, which is a good indication denoting that most studies have used non-toy applications.

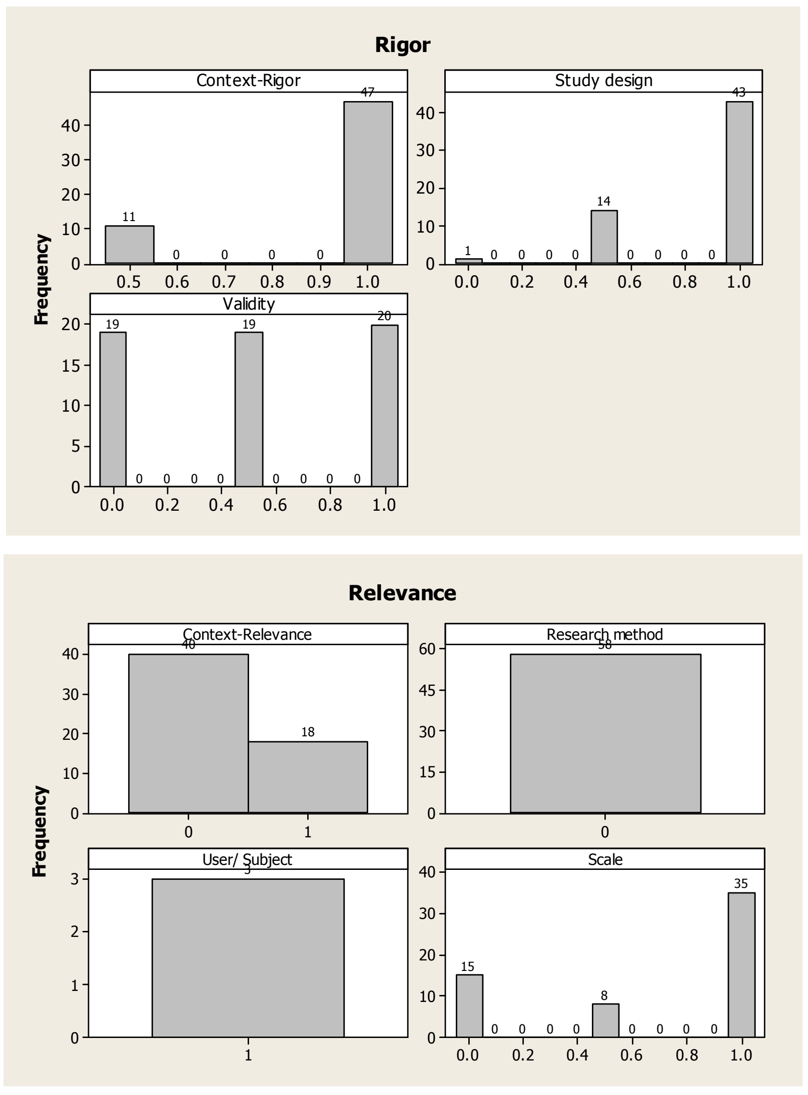


Figure 10- Histograms of Rigor and Relevance for the 58 empirical studies

Studies having highest level of rigor and industrial relevance are given in Table 20 and Table 21.

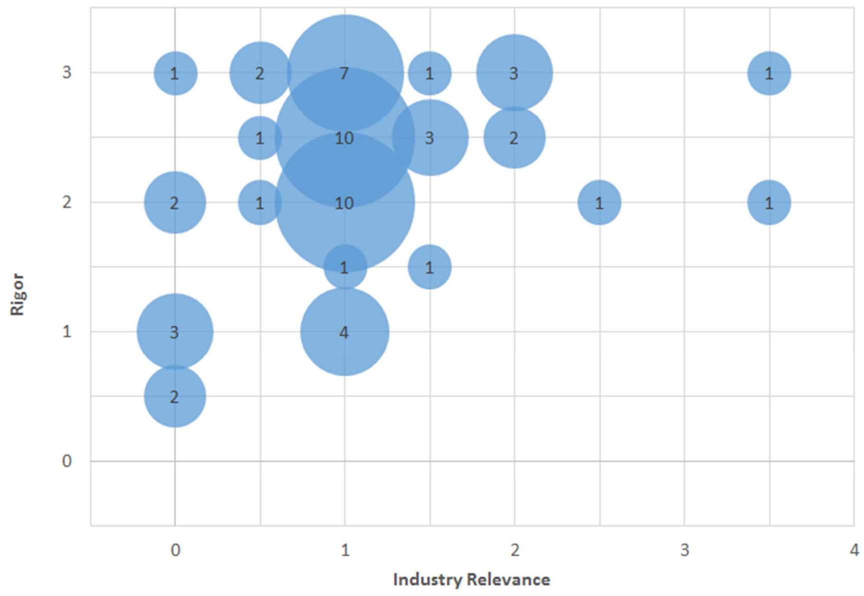
Table 20 Primary studies having top level of rigor

	Study	Level of Rigor			
		Context Described	Study Design Described	Validity Discussed	Total
1	S2	1	1	1	3
2	S9	1	1	1	3
3	S10	1	1	1	3
4	S12	1	1	1	3
5	S13	1	1	1	3
6	S17	1	1	1	3
7	S18	1	1	1	3
8	S19	1	1	1	3
9	S36	1	1	1	3
10	S38	1	1	1	3
11	S48	1	1	1	3
12	S49	1	1	1	3
13	S52	1	1	1	3
14	S65	1	1	1	3
15	S66	1	1	1	3
16	S70	1	1	1	3
17	S85	1	1	1	3
18	S90	1	1	1	3

Table 21 Top 10 primary studies having highest level of industrial relevance

	Study	Level of Industrial Relevance				
		Context	Research Method	User / Subject	Scale	Total
1	S76	1	0	1	1	3
2	S14	1	0	N/A	1	2
3	S34	1	0	N/A	1	2
4	S37	1	0	N/A	1	2
5	S49	1	0	0	1	2
6	S65	1	0	0	1	2
7	S70	1	0	0	1	2

8	S85	1	0	0	1	2
9	S90	1	0	0	1	2
10	S91	1	0	1	0	2



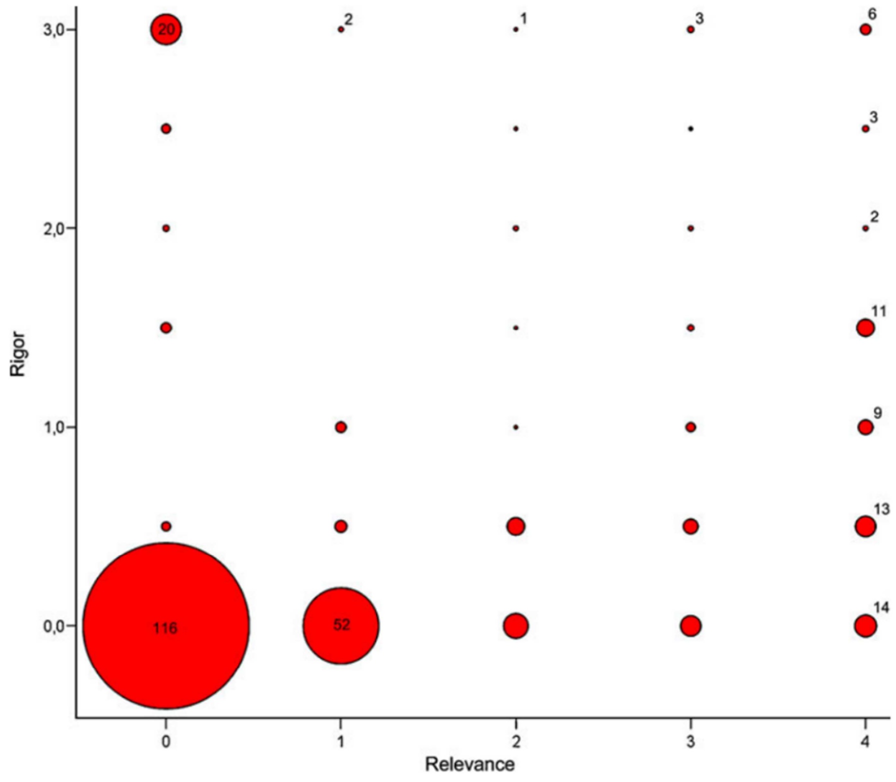


Figure 11- Rigor versus relevance of the empirical studies in this SLR versus [51]

Our next analysis was to assess the pair of rigor and relevance assessments for each pair, similar to what was done in [51]. The model was applied in [51] as part of a SLR of requirements engineering techniques in which 349 primary studies were analyzed. The pair-wise comparison of rigor and relevance in this SLR versus [51] is shown in Figure 11. A large portion of studies in [51], i.e., 116 articles, had zero rigor and relevance. This means that about one third of all the evaluations included in that SLR were experiments in which aspects related to rigor were not described or were application of a technology done by either students or researchers in academia in toy examples. However, since we only assessed the rubrics for papers with research facets of "validation research" and "evaluation research" in our study, the trends are better, in that we are observing quite a reasonable level of rigor and low to medium degree of relevance.

In addition to using the [51] to analyze the rigor of the studies, we also counted the number of RQs in each empirical study. From the top-cited guidelines on conducting empirical and case studies (e.g., [48]), it is evident that raising meaningful RQs for an empirical study will help better direct the study and the relevant measurements. Figure 12 shows the histogram of that data. 21 of the empirical studies in our pool did not raise any RQs. None had one RQ. A decreasing number of studies had between 2 and 5 RQs.

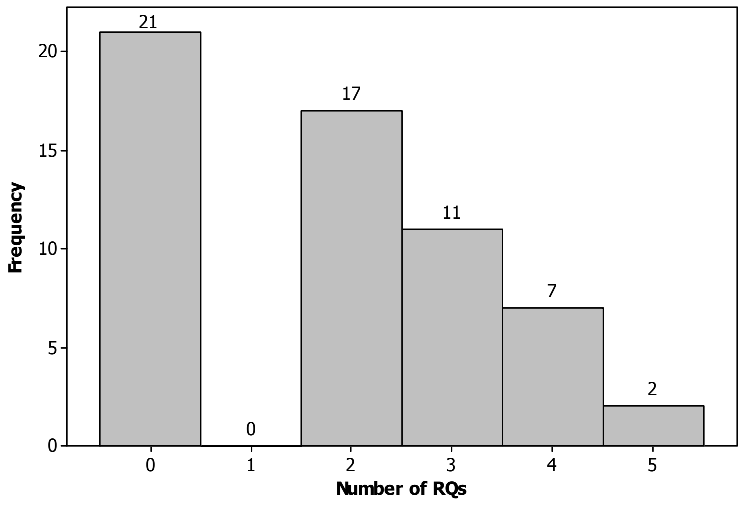


Figure 12- Number of RQs in empirical studies

By putting the results of RQs 2.2 and 2.3 together, similar to other SLRs (e.g., the one on search-based testing [13]), we can highlight the most frequently omitted aspects in the reporting of empirical studies in WAT. It is important that each empirical study is properly designed and reported so that it is easy to assess and replicate. Researchers should ensure to identify, report and address all relevant threats to validity of their studies and also aim at increasing the rigor and industrial relevance of the empirical studies as discussed above.

4.3. RQ 3-What is the state of empirical evidence in WAT?

We discuss results for RQ 3.1, 3.2, and 3.3 in the following.

4.3.1. RQ 3.1- Is there any evidence regarding the scalability of the WAT techniques?

Among the empirical studies, three [S18, S49, S68] explicitly studied scalability and reported the corresponding evidence. A summary of these works are shown in Table 22.

The study reported in [S18] presented results concerning the scalability of the approach, indicating that the performance of the algorithm will allow it to be used even in demanding scenarios such as those where daily regression testing is required. One of the two RQs in [S18] was: "Is the approach sufficiently computationally cheap to be applicable?" Authors raised the point that in order for the approach to be applicable, it must be possible to perform the entire repair process in a period of time that is commensurate with the time allocated to all other regression testing activities. As the results showed, the tool was able to complete its entire white-box analysis phase within three minutes for all of the applications considered. This suggested that the analysis phase of the algorithm and the tool that implements it are likely to have an acceptable performance, even for the most demanding web application regression testing scenarios. However, the four subject SUTs chosen in that study could only be considered medium scale (between 4 and 52 files with number of URLs between 14-150).

Table 22- Studies which empirically analyzed scalability issues

Study	Approach	Method to study scalability	Summary of results
S18	Automated Session Data Repair for Web Application Regression Testing	Measuring the execution time of the repair process for the four subject SUTs with different number of sessions (between 3-40).	The tool was able to complete its entire white-box analysis phase within three minutes for all of the applications considered.
S49	Invariant-based automatic testing of AJAX user interfaces	Measuring the execution time of the test tool ATUSA	The manual effort involved in setting up ATUSA (less than half an hour in the case study) is minimal. The scalability of the crawling and testing process is acceptable (it takes ATUSA less than 6 minutes to crawl and test TUDU, analyzing 332 clickables and detecting 34 states). The main component that can influence the performance and scalability is the crawling part. The performance of ATUSA in crawling an AJAX site depends on many factors such as the speed at which the server can handle requests, how fast the client-side JavaScript can update the interface, and the size of the DOM tree. ATUSA can scale to sites comprised of thousands of states easily.
S68	Using constraints on the inputs to reduce the number of transitions, thus compressing FSMs	Measuring the test model (FSM) size metrics, e.g., number of links and transitions	Both case studies show substantial savings by using the FSMWeb modeling technique over traditional FSMs for the modeling of web applications. These values lead to a 99.97% overall reduction in the number of states, and a 99.93% overall reduction in the number of transitions.

The study reported in [S49] presented an automatic testing approach for AJAX user interfaces based on invariants. In scalability analysis of the proposed test tool ATUSA, the authors found that the main component that can influence the performance and scalability is the crawling part. They then identified the factors impacting the performance of ATUSA in crawling an AJAX site, as shown in Table 22.

The study reported in [S68] used constraints on the inputs of a web application to reduce the number of transitions in its test model (FSM), thus compressing FSMs. To evaluate scalability, they measured the test model (FSM)'s size metrics, e.g., number of links and transitions. The reduction technique reduced the size of the FSMs significantly (see Table 22), thus helping the approach become more scalable.

4.3.2. RQ 3.2-Have different techniques been empirically compared with each other?

To answer this RQ and to be able to compare relevant techniques with one another, we utilized thematic analysis to group studies in relevant WAT areas together. We followed the thematic analysis guidelines [40-42] and developed a set of ten WAT theme areas, as shown in Table 23. Frequency of studies under each WAT theme area and the trend of the number of studies under each area are shown in Table 23 and Figure 13, respectively. Note that each paper was only classified under one theme area in this case. Each paper was classified under the theme area which was the most applicable for it.

Table 23- Frequency of empirical studies under each WAT theme area

Theme area	# of papers	Percentage	References
White box	7	12%	S19, S28, S33, S40, S45, S47, S62
Black box, FSM, model-based	5	9%	S3, S47, S68, S77, S90
Mutation	3	5%	S13, S36, S70

AJAX testing	6	10%	S2, S48, S49, S66, S69, S85
Session-based testing (navigation models, logs)	16	28%	S4, S10, S12, S17, S18, S24, S37, S43, S51, S52, S54, S65, S67, S83, S84, S93
Cross-Browser Compatibility Testing	3	5%	S14, S30, S92
Regression testing	5	9%	S9, S18, S21, S50, S66
Oracle	4	7%	S16, S17, S22, S95
Support for testing	9	16%	S5, S29, S38, S46, S57, S59, S61, S64, S76
Other	4	7%	S15, S34, S63, S91

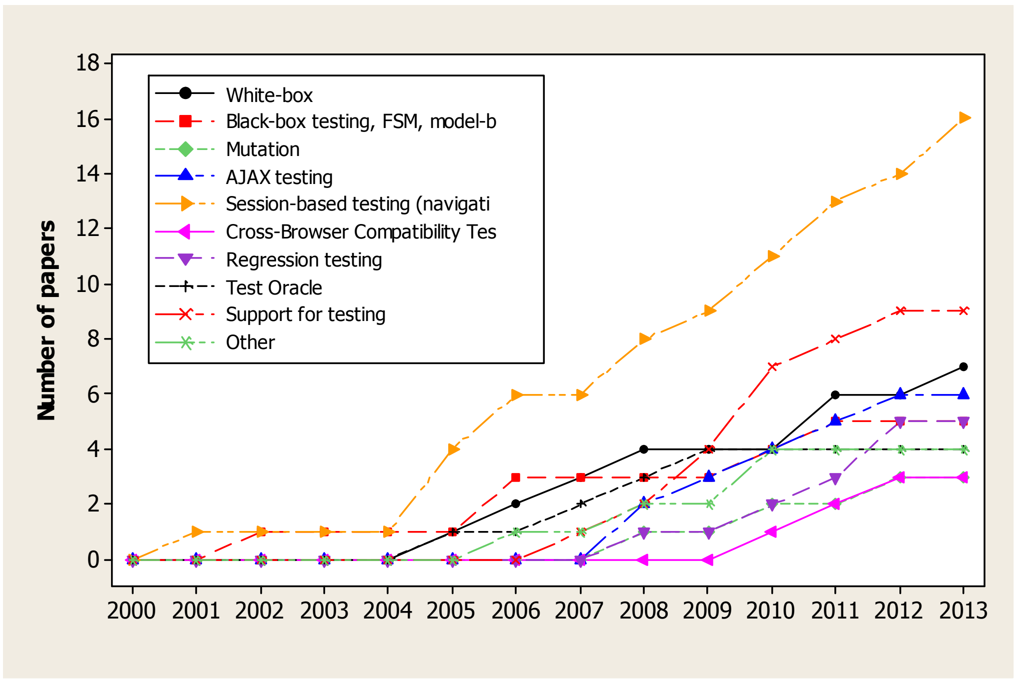


Figure 13- Trend of the cumulative number of studies under each WAT theme area over the years

To answer the RQ, using the above theme areas, we extracted the list of studies which have explicitly conducted empirical comparisons with other studies. We found 11 such studies (Table 24). The studies have been divided into the WAT theme areas and we synthesize next the empirical comparisons under each theme area.

Table 24- List of studies conducting empirical comparisons with other studies/tools

Study	**Year of publication**	**Compared studies/tools**
White-box testing		
[S40]	2008	[S45, S72]
Session-based testing		
[S10]	2005	[S12], [52]
[S24]	2009	[S12], [52]
[S47]	2006	[52]
[S52]	2005	[S11]
[S65]	2008	[S12, S16, S17]
[S83]	2006	[S54]
[S93]	2013	A web scraping/crawling framework called Scrappy (www.scrapy.org)
AJAX testing		
[S2]	2008	[S1, S69, S81]
Cross-Browser Compatibility Testing		
[S30]	2012	[S92, S14]
Support for testing		
[S64]	2009	[S45][53], and a crawler called Spider (www.owasp.org)

4.3.2.1. White-box testing

There was one paper, [S40], which conducted comparative empirical study in the theme area of white-box testing. The authors of [S40] proposed a dynamic test generation technique, based on combined concrete and symbolic execution, for PHP applications. The authors compared their technique and tool (called Apollo) to two other approaches. First, they implemented an approach similar to [S45] for JavaScript testing (referred to as Randomized). Second, they compared their results to those reported by a static analysis technique [S72], on the same subject programs.

Apollo test generation strategy outperformed the randomized testing (proposed by [S45]) by achieving an average line coverage of 58.0%, versus 15.2% for Randomized. The Apollo strategy significantly outperformed the Randomized strategy by finding a total of 214 faults in the subject applications, versus a total of 59 faults for Randomized.

For the three overlapping subject programs, Apollo was both more effective and more efficient than the tool presented in [S72]. Apollo found 2.7 times as many HTML validation faults found by [S72]'s tool (120 vs. 45). Apollo found 83 execution faults, which are out of reach for [S72]'s tool. Apollo is also more scalable on schoolmate, the largest of the programs, Apollo found 104 faults in 10 minutes, while [S72]'s tool found only 14 faults in 126 minutes. The authors discussed that the time spent in [S72]'s tool is due to constructing large automata and to the expensive algorithm for checking disjointness between regular expressions and context-free languages.

4.3.2.2. Session-based testing

There were seven papers which conduced comparative empirical studies in the theme area of session-based testing, which are discussed next.

The study [S10] was an empirical comparison of three test-suite reduction techniques for user-session-based testing: (1) concept analysis [S12], (2) the HGS requirements-based approach [52], and (3) the Greedy requirements-based approach [52]. Note that the paper [52] is not in our pool, since it is not a WAT-specific paper, but rather is in the general area of software testing. They compared the reduced test suite size, program coverage, fault detection, and time and space costs of each of the techniques for two web applications. The results showed that concept analysis-based reduction is a cost-effective alternative to requirements-based approaches.

The study [S24] presents a user-session based testing technique that clusters user sessions based on the service profile and selects a set of representative user sessions from each cluster. Two empirical studies are then presented to compare the proposed approach with the approaches reported in S12 (based on concept analysis) and [52] (based on URL coverage). The results demonstrated that the proposed approach consistently detected the majority of the known faults by using a relatively small number of test cases in both studies, and performed better than the two other approaches.

In [S47], the authors explored three different strategies to integrate the use of coverage-based requirements and usage-based requirements in relation to test suite reduction for web applications. They investigated the use of usage-based test requirements for comparison of test suites that have been reduced based on program coverage-based test requirements. They examined the effectiveness of a test suite reduction process based on a combination of both usage-based and program coverage-based requirements. Finally, they modified a popular test-suite reduction algorithm [52] to replace part of its test selection process with selection based on usage-based test requirements. The case study results suggested that integrating program coverage-based and usage-based test requirements has a positive impact on the effectiveness of the resulting test suites.

The study [S52] reports a comprehensive study comparing seven user-session-based test techniques: (white box techniques) (1) WB-1 (white box); the simplest implementation by Ricca and Tonella [S11], (2) WB-2: WB-1 with boundary values, (User-session based techniques) (3) US-1: Direct reuse of user sessions, (4) US-2: Combining different user sessions, (5) US-3: Reusing user sessions with form modifications, and (Hybrid approaches) (6) HYB-1: Partially satisfying testing requirements with user session data, and (7) HYB-2: Satisfying testing requirements with user session data and tester input. The empirical study of [S52] found that WB-2 provided the greatest code coverage with 76% and 99% of the blocks and functions covered, respectively. US-3 provided the greatest fault detection capabilities with 63% of the faults detected. An important finding from this paper is that user session data can be used to produce test suites more effective overall than those produced by the white-box techniques considered; however, the faults detected by the two classes of techniques differ, suggesting that the techniques are complementary.

The study [S65] proposed several new test suite prioritization strategies for web applications and examined whether these strategies can improve the rate of fault detection for three web applications and their preexisting test suites. Experimental results show that the proposed prioritization criteria often improve the rate of fault detection of the test suites when compared to random ordering of test cases. [S65] borrowed the experimental objects and methodology from [S12, S16, S17]. The goal of empirical study in [S65] was to assess the effectiveness of the prioritization strategies by evaluating their fault detection rate, i.e., finding the most faults in the earlier tests. The study concluded that none of their prioritization criteria is clearly the "best criteria" for all cases, but depending on the tester's goal and the characteristics of the web applications, different prioritization strategies may be useful. Specific guidelines were given in this regard.

The study [S83] describes two experiments that replicated Kallepalli and Tian's work [S54], which had used Unified Markov Models (UMMs) as usage-based statistical model, built from Web server access logs, as basis for test case selection. In addition, server error logs were also used to measure a Web application's reliability and consequently to investigate the effectiveness of UMMs as a suitable testing mechanism. Their results showed that, in contrast to findings of [S54], multiple set UMMs were needed for trustworthy test case generation. In addition, the reliability assessment reported in [S83] corroborated results from [8], confirming that UMMs seem to be a suitable testing mechanism to use for testing web applications.

The study [S93] presented an approach and a prototype tool called WebMate which systematically explores and tests all distinct functions of a web application. The prototype tool handles interfaces as complex as Facebook and is able to cover up to 7 times as much code as existing tools. The only requirements to use WebMate are the URL of the application and, if necessary, user name and password. The empirical study reported in [S93] compared the performance of WebMate versus a popular open-source web scraping framework for Python called Scrappy. When compared to alternative tool Scrappy, WebMate achieved higher coverage (seven times better for one SUT and four times better for 2 other SUTs).

4.3.2.3. AJAX testing

Out of the six studies focusing on AJAX testing [S2, S48, S49, S66, S69, S85], the only comparative study that we found in the theme area of AJAX testing is [S2]. The authors of [S2] had proposed a state-based testing technique for AJAX in an earlier work [S69]. [S2] reported a case study-based comparison of four types of testing techniques applied to AJAX web applications: (1) state-based testing [S69], (2) coverage-based/white-box testing [S1], (3) black-box (record and playback tools), and (4) UML model-based testing [S81]. The study was quite rigorous in its comparative analysis as it explicitly followed a systematic approach for comparing efficiency, effectiveness and applicability of testing techniques [54]. The study found that state-based testing is complementary to the existing AJAX testing techniques and can reveal faults otherwise unnoticed or hard to reveal with the other techniques. Also, the study reported that, overall, state-based testing involves more effort than the other considered techniques; hence, there is a trade-off between fault revealing potential and effort involved.

4.3.2.4. Cross-Browser Compatibility Testing

Out of the three studies focusing on cross-browser compatibility testing [S14, S30, S92], there was one comparative study [S30] which compared its proposed tool (called CrossCheck) to the two tools (CrossT and WebDiff) proposed in the two earlier studies [S92, S14].

As the metrics used for comparisons, the study measured trace-level (TL) and screen-level (SL) cross-browser compatibility differences. Executed on six open-source case-study systems, CrossCheck outperformed CrossT and WebDiff. The improvement over CrossT was attributed to a better screen-level matching in CrossCheck, whereas the improvement over WebDiff is due to the use of the machine learnt classifier for visual comparison.

4.3.2.5. Support for testing

Among the nine studies which were classified under the "support for testing" theme area, one [S64] conducted a comparative study. This study proposed an approach for precise interface identification to improve testing and analysis of web applications. The empirical study was conducted to assess the efficiency, precision, and usefulness of the approach for interface identification (named *wam-se*). To do so, the authors compared *wam-se* against three other approaches: *wam-df* [S45], *dfw* [53], and a tool called Spider. In the empirical evaluation, the authors show that the set of interfaces identified by their approach is more accurate than those identified by other approaches. They also showed that this increased accuracy would lead to improvements in several important quality assurance techniques for web applications: test-input generation, penetration testing, and invocation verification.

4.3.3. RQ 3.3-How much empirical evidence exists for each category of techniques and type of web apps?

To address this question, we examined the distribution of empirical studies on their target type of web applications and the target technologies.

We considered the following attributes to categorize a web application: location of the application under test (server side or/and client side), dynamicity (static or dynamic pages), and synchronicity of HTTP calls (asynchronous e.g. Ajax, or synchronous). Figure 14 shows the results for each attribute and Table 25 shows the combined picture.

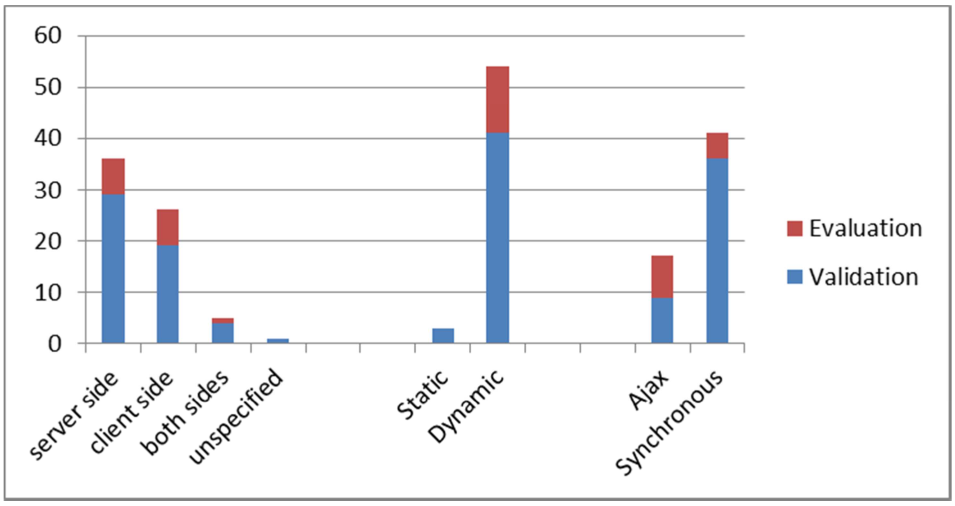


Figure 14- Empirical studies break down on the attributes of web applications under test: location, dynamicity, and synchronicity.

As Figure 14 shows, most of the empirical studies (62% of the studies, 36 papers) target the server side applications, 26 studies (45%) target client side applications and only 5 studies (9%) provide empirical results on both sides of a web application. As for the dynamicity, the figure shows that the majority of the studies generated empirical evidence on testing dynamic web applications (54 papers, 93%) and only 3 papers aim for testing static web applications. Finally, Figure 14 shows that only 17 studies focus on asynchronous HTTP communication (Ajax). Since implementation of such communication is highly related to client side of a web application, we considered the 26 studies that investigated testing on the client side applications. 65% of these studies provided empirical evidence on Ajax applications.

Table 25- Frequency of studies under each WAT theme area

	Dynamic Application		**Static Application**	
	Validation	**Evaluation**	**Validation**	**Evaluation**
Server side	S9, S12, S15, S16, S17, S18, S21, S22, S24, S28, S33, S37, S40, S43, S45, S47, S51, S63, S64, S68, S76, S77, S83, S84, S94	S4, S10, S19, S52, S65, S90	S83	--

Client side	Asynchronous (Ajax)	S3, S14, S34, S43, S50, S67, S69, S92, S93	--	--	--
	Synchronous	S5, S30, S54, S59, S61, S62, S91	S2, S36, S49, S57, S66, S85	S54, S91	--
Both	Asynchronous (Ajax)	--	--	--	--
	Synchronous	S13, S29, S38, S46	S48	--	--
Unspecified		S70		--	--

We also investigated how much evidence exists on the client-tier technology and on the server-tier technology. Table 26 shows the distribution of the 26 client side testing studies. The table shows that the most of the evidence, as expected, is on HTML with 26 studies. There are 13 empirical studies on JavaScript and 12 studies on DOM technologies.

Table 26- Empirical studies presenting results on Client-tier web technologies

	HTML	**DOM**	**JavaScript**
Validation	S3, S13, S16, S17, S24, S28, S29, S38, S43, S46, S50, S51, S54, S61, S62, S70, S83, S91, S92, S93, S94	S14, S30, S34, S43, S50, S59, S61, S62, S67, S69, S70, S92	S3, S5, S14, S29, S30, S34, S38, S46, S50, S59, S69, S92, S93
Evaluation	S2, S19, S49, S57	0	0

Table 27 shows the distribution of the 36 studies that provided empirical results on the server side. Most of the evidence we have is on the J2EE technology (21 studies, 58%), followed by the PHP technology with 12 studies. There are two studies on .Net framework, two studies on Perl/CGI technology, and one study on LISP [S85].

Table 27- Empirical studies presenting results on Server-tier web technologies

	PHP	**J2EE**	**.Net**	**Perl/CGI**	**Other**
Validation	S9, S21, S29, S37, S38, S40, S63, S70,S92	S12, S13, S15, S16, S17, S24, S28, S43, S47, S51, S64, S67, S69, S76, S84	S33, S38	S22	-
Evaluation	S10, S19, S49	S4, S10, S49, S65, S85,S90	-	S52	S85

CHAPTER 5

DISCUSSIONS

Summarized discussions and findings of this study along with potential threats to validity are presented in this section.

5.1. Findings, Trends

We summarize and discuss findings and trends for each of the RQs next;

RQ 1.1-Types of input/inferred test models: The navigation models seem to be the most popular as 22 studies used a type a of navigation models. Examples include finite-state machines (FSM) which specify the flow among the pages of a web application. Control and data flow models in unit level and also DOM models have also been used in several studies. Other types of models such as: program dependence graphs (PDGs) and Database Extended Finite State Machine (DEFSM) have also been proposed.

RQ 1.2-Types of fault models/bug taxonomy: 21 studies discussed fault models specific to web applications. Over 50 types of faults (e.g., faults related to browser incompatibility, and faults in session synchronization) have been discussed. Test techniques targeting some of these fault types have been proposed.

RQ 1.3-Tools and their capabilities: 52 of the 95 papers (54%) presented (mostly prototype-level) tool support for the proposed WAT approaches. Only 11 of the 52 presented tools (21%) were available for download. We noticed that in the papers presented after 2008, more and more tools are available for download, which is a good sign for the community. We extracted the features and capabilities of the tools available for download.

RQ 2.1-Metrics used for assessing cost and effectiveness: There have been four types of cost metrics used in the empirical studies in this area: (1) effort/test time, (2) test-suite size, (3) memory space, and (4) other. Measuring test effort/time was the most frequent. We categorized the effectiveness metrics as the following: (1) code coverage, (2) model or requirements coverage, (3) other types of coverage, (4) detecting real faults, (5) detecting injected faults, and (6) other metrics such as number of DOM violations or reliability growth. Code coverage was the most frequent metric in this category.

RQ 2.2-Threats to validity in the empirical studies: external and internal validity threats are the mostly addressed threats, in 36 and 30 studies, respectively. Construct and conclusion threats were identified in 9 and 6 studies only. Representativeness of injected faults/mutations was the most frequent identified type of internal validity threats (in 17 studies). Representativeness of SUTs was the most frequent identified type of external validity threats (in 32 studies). Examples of construct validity threats identified in the studies are: subjectivity in bug classification, and not considering the severity of the faults. Examples of conclusion validity threats identified in the studies are: not considering all types of effort spent, results relying on interpretation of metrics, and results relying on human based classification.

RQ 2.3-The level of rigor and industrial relevance: In most of the empirical studies (47 of the 58), the context of the empirical study has described to a degree where a reader can understand and compare it to another context. Study design has been also explained well in most of the studies (43 of the 58). 40 of the 58 studies were performed in a laboratory setting, while 18 were conducted in industrial context or on an industrial real web application. As long as the research methods are concerned, none of the studies used methods relevant for practitioners, e.g., action research.

RQ 3.1-Evidence regarding the scalability: Among the empirical studies, three of them explicitly studied scalability and reported the corresponding evidence. The methods they used to study scalability were as follows: measuring the execution time of the repair process for the subject SUTs with different number of sessions, measuring the execution time of the test tool, measuring the test model (FSM) size metrics, e.g., number of links and transitions.

RQ 3.2-Empirical comparison of techniques: We found 11 studies which have conducted empirical comparisons with other studies/tools. We divided those papers into the WAT theme areas and we synthesized the empirical comparisons under each theme area.

RQ 3.3-Empirical evidence for each category of techniques: Most of the empirical studies (62% of the studies, 36 papers) target the server side, 26 studies (45%) target the client side and only 5 studies (9%) provide empirical results on both sides of a web application. As for the dynamicity, the majority of the studies reported empirical evidence on testing dynamic web applications (54 papers, 93%) and only 3 papers aim for testing static web applications. 17 studies focus on asynchronous HTTP communication (Ajax). 65% of these 26 studies provided empirical evidence on Ajax applications. When we consider the client-tier technology, most of the evidence, as expected, is on HTML with 26 studies (all client-side studies), 13 empirical studies on JavaScript and 12 studies on DOM technologies. On the other hand, in server-side studies, most of the evidence is on the J2EE technology (21 studies), followed by the PHP technology with 12 studies.

5.2. Discussion on Validity Threats

The results of a SLR can be affected by a number of factors such as the researchers conducting the study, the data sources selected, the search term, the chosen time-frame, and the pool of primary studies. Below we discuss potential threats to validity of this study and the steps we have taken to mitigate or minimize them.

5.2.1. Internal Validity

One threat could be incomplete selection of publications. We presented in Section 3.3 a detailed discussion around the concrete search terms and the databases used in our study. In order to obtain a complete set of primary studies covering the given research topic as possible, the search term was derived systematically. Different terms for web application testing and analysis were determined with many alternatives and different combinations. However, the list might not be complete and additional or alternative terms might have affected the number of papers found.

Furthermore, our inclusion and exclusion criteria were discussed in Section 3.3.2. The decision on which papers to include in the final pool depended on the group judgment of the researchers conducting the SLR. As discussed in Section 3, the authors adopted a defined systematic voting process among the team in the paper selection phase for deciding whether to keep or exclude any of the papers in the first version of the pool. This process was also carried out to minimize personal bias of each of the authors. When the authors of the study disagreed, discussions took place until an agreement was reached. A high conformance value was achieved, which indicates a similar understanding of relevance.

Though a replication of this SLR may lead to a slightly different set of primary studies, we believe the main conclusions drawn from the identified set of papers should not deviate from our findings.

5.2.2. Construct Validity

Construct validity is concerned with the extent to what was to be measured was actually measured. In other words, threats to construct validity refer to the extent to which the study setting actually reflects the construct under study. As discussed, based on the classification scheme developed in the earlier SM study [2], after the papers in the pool were systematically mapped, the actual data extraction and synthesis took place. The pool of papers was partitioned among the authors. Each author first extracted the data by mapping the paper inside the classification scheme and also extracting the evidence and empirical aspects of each paper independently. Then a systematic peer review process among the authors was conducted in which the data and attributes extracted by each researcher were cross-checked by another researcher. In case of differences in opinions, online discussions (e.g., email, Skype) were conducted to resolve the differences. This cross-check helped the team to extract the data and conduct the measurement in a reliable manner. The above steps mitigate some of the threats to construct validity of our study.

5.2.3. Conclusion Validity

It is important for a SLR study to present results and conclusions that are directly traceable to data and results that have in turn been carefully extracted from the primary studies, and can be reproduced by other researchers. To ensure conclusion validity of our study, we presented throughout the Section 4 graphs generated directly from the data and discussed the explicit observations and trends based on synthesis of those data. This ensures a high degree of traceability between the data and conclusions. Furthermore, to ensure traceability of the extracted data, evidence and synthesis, the entire raw data of the SM and the SLR are available online in the form of a spreadsheet in the Google Docs system [33]. This will enable transparency and also replicability of our analysis.

5.2.4. External Validity

The results of the SLR study were considered with respect to approaches in the software engineering domain. Thus, the data and findings presented and the conclusions drawn are only valid in the given context (web application testing). Additional papers and approaches that are identified in the future can be categorized and synthesized accordingly. Due to the systematic procedure followed during the SLR study, we believe our study is repeatable.

CHAPTER 6

CONCLUSIONS & FUTURE WORK

6.1. Conclusions

The web has proven to be a powerful medium for delivering software services over the Internet. Due to its inherited distributed complexity and dynamism, testing is known to be a challenge for web developers. That is why many researchers have worked in this domain from the early days of the web.

In this thesis study, we presented first SLR in the domain of web application functional testing (WAT), targeting the studies published between 2000 and 2013. Our initial search retrieved 193 papers of which 95 were included in this study using a selection strategy. This is as a follow-up complementary study of a recent SM study conducted by V. Garousi et al.

The complete process that we have used in this study; consisting of article selection strategy, article voting for inclusion and exclusion, identification of research questions, identification of attributes for data extraction, data extraction and synthesis, has been described in more detail to assist similar future researches. In order to provide more transparent, reproducible and extensible study, the primary studies that we have used in our pool were uploaded to Google Docs system and made available to public access. Also the whole extracted data, our comments and discussions are available online in Google Docs as a spreadsheet [33]. Our publicly available online repository of studies in WAT domain and Google Docs study environment would be a source of inspiration to further similar SLR studies.

Our study indicates that web testing is an active area of research with an increasing number of publications. Among other results, we synthesized the following data/findings from the papers to answer our RQs: (1) the types of input/inferred test models, (2) the fault models/bug taxonomy related to web applications, (3) test tools proposed in this area and their capabilities, (4) metrics used for assessing cost and effectiveness of WAT techniques, (5) the threats to validity in the empirical studies, (6) level of rigor and industrial relevance of the empirical studies, (7) evidence regarding the scalability of the WAT techniques, (8) empirical comparisons in WAT domain and (9) amount of empirical evidence for each category of techniques and type of web applications. Our SLR shows the state-of-the-art in web application testing, areas that have been covered and techniques/tools that have been proposed. It provides a guideline to assist researchers in planning future work by analyzing the existing evidence for different WAT techniques and their effectiveness and also by spotting research areas that need more attention.

Identified types of input/inferred test models which were used or proposed in the WAT studies showed that the navigation models are the most preferred models followed by DOM models. These two models especially used in system and integration level testing. Control and data flow models are the next popular models mostly used in unit level testing with the aim of modeling the flow inside a function or module.

We extracted the features and capabilities of the WAT tools presented in the studies which will contribute future researches by enabling access to the information of all proposed tools in WAT domain in single point. 54% of the studies presented (mostly prototype-level) new tool for the proposed WAT approaches. However, only 21% of the presented tools were available for download. In order to improve the contribution to the community with presented tools and enable improvement of the tool by other researchers and industry, making both the executable and source code of the tools available for download is important. When we checked the trend among the years, we noticed that after 2008, more and more tools are available for download, which is a good sign for the community.

"Threats to validity" is one of the important topics that we have analyzed and discussed in this study. In order extract the threat data from the studies, we created a classification using descriptions about the causes of threats. This classification can contribute future studies that need a WAT threat classification or researchers could benefit for identifying their validity threats. The overall picture showed us that the mostly addressed validity threats are "representativeness of SUTs" as external and "representativeness of injected faults/mutations" as internal validity threat. The sum of these two corresponds to almost half of the total number of all threats. From this point forward, it is easy to say that these two are the main threats to validity of WAT studies.

In most of the empirical studies (81%), the context of the empirical study has described to a degree where a reader can understand and compare it to another context. Study design has been also explained well in most of the studies (74%). However, only 22 out of 58 studies described the validity threats in enough detail. Validity discussion seems to be the most omitted aspect in empirical studies which is an important argument for evaluating the level of rigor. On the other hand, when we focus on industrial relevance, 69% of the empirical studies were performed in a laboratory setting, while remaining few were conducted in industrial context or on an industrial real web application. When the research methods are concerned, none of the studies used methods relevant for practitioners, e.g., action research. %60 of the studies used relatively large scale SUTs which is the only one good indicator for industrial relevance. These results show us that in most of the empirical studies, potential of impact on industry was not considered or sufficient attention was not shown, which is a very important factor for determining the success of a study in this applied research field.

Among the empirical studies, three of them explicitly studied scalability and reported the corresponding evidence. Scalability was also identified as a type of external validity threat. Most of the studies mentioned "Representativeness of SUTs" threat considering the small size and low complexity of SUTs which also corresponds to scalability. But almost none of these studies have measured the scalability of their approach. We also faced the scalability factor while analyzing the industrial relevance. The context of the study and the scale of the applications used in the evaluations are closely related to scalability issue and are important arguments for evaluating the industrial relevance. All these findings showed us that, evidences regarding the scalability of approaches are not sufficient and measurement of the scalability is an omitted aspect of empirical WAT studies.

6.2. Future Work

When we look at the types of fault models and bug taxonomies specific to web applications, we see that over 50 types of faults have been discussed. Test techniques targeting some of these fault types have been proposed. It is worth conducting more in-depth studies in future to ensure coverage of all the fault types by the test techniques and also the effectiveness of those techniques on detecting each specific fault type.

We have highlighted the current state of model usage in WAT studies. Further studies could be conducted for deeply evaluating the capabilities, advantages and disadvantages of each distinct model in specific WAT problems.

As we identified two types of validity threats were the mostly addressed ones in WAT studies. Conducting further studies targeting the minimization of these two main threats would bring a great impact on the validity of WAT studies.

According to our findings, industrial relevance is the one of the most omitted aspects in empirical WAT studies. However, this is a very important factor for the success of a study in this applied research field. It is obvious that this aspect of the studies requires more attention for impacting the industry.

As we discussed that evidences regarding the scalability of approaches are not sufficient and measurement of the scalability is an omitted aspect of empirical WAT studies. Further attention to scalability of approaches would either enhance the external validity or the industrial relevance of empirical studies.

As this study focus on functional WAT, a future SM or SLR study can be conducted for non-functional side of WAT (e.g. security, scalability, performance, usability…) using our process and similar research questions.

REFERENCES

References of Primary Studies

[S1] P. Tonella and F. Ricca, "A 2-layer model for the white-box testing of Web applications," IEEE International Workshop on Web Site Evolution, pp. 11–19, 2004

[S2] A. Marchetto, F. Ricca, and P. Tonella, "A case study-based comparison of web testing techniques applied to AJAX web applications," International Journal on Software Tools for Technology Transfer, vol. 10, no. 6, pp. 477–492, Oct. 2008

[S3] S. Artzi, J. Dolby, S. H. Jensen, A. Moller, and F. Tip, "A framework for automated testing of JavaScript web applications," International Conference on Software Engineering, pp. 571–580, 2011

[S4] S. Sprenkle, L. Pollock, and L. Simko, "A Study of Usage-Based Navigation Models and Generated Abstract Test Cases for Web Applications," IEEE International Conference on Software Testing Verification and Validation, pp. 230–239, 2011

[S5] P. Saxena, D. Akhawe, S. Hanna, F. Mao, S. McCamant, and D. Song, "A Symbolic Execution Framework for JavaScript," IEEE Symposium on Security and Privacy, pp. 513–528, 2010

[S6] G. a Di Lucca, a R. Fasolino, and P. Tramontana, "A Technique for Reducing User Session Data Sets in Web Application Testing," IEEE International Symposium on Web Site Evolution, pp. 7–13, 2006

[S7] A.M. Törsel, "A Tool for Web Applications Using a Domain-Specific Modelling Language and the NuSMV Model Checker", IEEE International Conference on Software Testing, Verification and Validation, 2013

[S8] Y. Qi, D. Kung, and E. Wong, "An agent-based data-flow testing approach for Web applications," Information and Software Technology, vol. 48, no. 12, pp. 1159–1171, 2006

[S9] A. Marback, H. Do, and N. Ehresmann, "An Effective Regression Testing Approach for PHP Web Applications," IEEE International Conference on Verification and Validation, pp. 221-230, 2012

[S10] S. Sprenkle, S. S. S. Sampath, E. Gibson, L. Pollock, and a Souter, "An empirical comparison of test suite reduction techniques for user-session-based testing of Web applications," IEEE International Conference on Software Maintenance, pp. 587–596, 2005.

[S11] F. Ricca and P. Tonella, "Analysis and testing of Web applications," Proceedings of the 23rd International Conference on Software Engineering, vol. 47, no. 6, pp. 25–34, 2001.

[S12] S. Sampath, S. Sprenkle, E. Gibson, L. Pollock, and a S. Greenwald, "Applying Concept Analysis to User-Session-Based Testing of Web Applications," IEEE Transactions on Software Engineering, vol. 33, no. 10, pp. 643–658, 2007.

[S13] U. Praphamontripong and J. Offutt, "Applying Mutation Testing to Web Applications," International Conference on Software Testing Verification and Validation Workshops, pp. 132–141, 2010

[S14] A. Mesbah and M. R. Prasad, "Automated cross-browser compatibility testing," International Conference on Software Engineering, pp. 561–570, 2011

[S15] W. G. J. Halfond and A. Orso, "Automated identification of parameter mismatches in web applications," Proceedings of the ACM SIGSOFT International Symposium on Foundations of software engineering, p. 181, 2008

[S16] S. Sprenkle, L. Pollock, H. Esquivel, B. Hazelwood, and S. Ecott, "Automated Oracle Comparators for Testing Web Applications." IEEE International Symposium on Software Reliability, pp. 117-126, 2007

[S17] S. Sprenkle, E. Gibson, S. Sampath, and L. Pollock, "Automated replay and failure detection for web applications," Proceedings of the IEEEACM international Conference on Automated software engineering, pp. 253–262, 2005

[S18] M. Harman and N. Alshahwan, "Automated Session Data Repair for Web Application Regression Testing," 2008 International Conference on Software Testing, Verification, and Validation, pp. 298–307, 2008

[S19] N. Alshahwan and M. Harman, "Automated web application testing using search based software engineering," IEEEACM International Conference on Automated Software Engineering, pp. 3–12, 2011.

[S20] M. H. Alalfi, J. R. Cordy, and T. R. Dean, "Automating Coverage Metrics for Dynamic Web Applications," European Conference on Software Maintenance and Reengineering, pp. 51–60, 2010.

[S21] K. Dobolyi, E. Soechting, and W. Weimer, "Automating regression testing using web-based application similarities," International Journal on Software Tools for Technology Transfer, vol. 13, no. 2, pp. 111–129, 2010.

[S22] L. Ran, C. Dyreson, A. Andrews, R. Bryce, and C. Mallery, "Building test cases and oracles to automate the testing of web database applications," Information and Software Technology, vol. 51, no. 2, pp. 460–477, 2009

[S23] J. Offutt, Y. Wu, X. Du, and H. Huang, "Bypass testing of Web applications," International Symposium on Software Reliability Engineering, pp. 187–197, 2004.

[S24] X. Luo, F. Ping, and M.-H. Chen, "Clustering and Tailoring User Session Data for Testing Web Applications," International Conference on Software Testing Verification and Validation, pp. 336–345, 2009

[S25] S. Sampath, V. Mihaylov, A. Souter, and L. Pollock, "Composing a Framework to Automate Testing of Operational Web-Based Software," Proceedings of the 20th IEEE International Conference on Software Maintenance, pp. 104-113, 2004

[S26] W. I. C. C. Hu, "Constructing an Object-Oriented Architecture for Web Application Testing," J. Inf. Sci. Eng. 18, no. 1, pp. 59–84, 2002.

[S27] F. Ricca and P. Tonella, "Construction of the system dependence graph for Web application slicing," Proceedings IEEE International Workshop on Source Code Analysis and Manipulation, pp. 123–132, 2002.

[S28] S. Sampath, E. Gibson, S. Sprenkle, and L. Pollock, "Coverage Criteria for Testing Web Applications," Computer and Information Sciences, University of Delaware, Tech. Rep, 2005-017, 2005

[S29] N. Alshahwan, M. Harman, A. Marchetto, R. Tiella, P. Tonella, and F. B. Kessler, "Crawlability Metrics for Web Applications," IEEE International Conference on Software Testing, Verification and Validation, pp. 151-160, 2012.

[S30] S. R. Choudhary, M. R. Prasad, and A. Orso, "Crosscheck: Combining crawling and differencing to better detect cross-browser incompatibilities in web applications". International Conference on Software Testing, Verification and Validation, pp. 171-180, Apr.2012.

[S31] C.-H. Liu, "Data flow analysis and testing of JSP-based Web applications," Information and Software Technology, vol. 48, no. 12, pp. 1137–1147, Dec. 2006.

[S32] E. Di Sciascio, F. M. Donini, M. Mongiello, R. Totaro, and D. Castelluccia, "Design Verification of Web Applications Using Symbolic Model Checking," Web Engineering. Springer Berlin Heidelberg, pp. 69–74, 2005.

[S33] M. Ozkinaci and A. Betin Can, "Detecting Execution and HTML Errors in ASP .Net Web Applications," Proceedings of the 6th International Conference on Software and Data Technologies, pp. 172–178, 2011.

[S34] K. Pattabiraman and B. Zorn, "DoDOM: Leveraging DOM Invariants for Web 2.0 Application Robustness Testing," IEEE International Symposium on Software Reliability Engineering, pp. 191–200, Nov. 2010.

[S35] P. Tonella and F. Ricca, "Dynamic model extraction and statistical analysis of Web applications," Proceedings International Workshop on Web Site Evolution, pp. 43–52, 2002.

[S36] S. Mirshokraie, A. Mesbah and K. Pattabiraman, "Efficient JavaScript Mutation Testing," IEEE International Conference on Software Testing, Verification and Validation, 2013.

[S37] T. Ettema and C. Bunch, "Eliminating Navigation Errors in Web Applications via Model Checking and Runtime Enforcement of Navigation State Machines," Proceedings of the IEEE/ACM international conference on Automated software engineering ACM, pp. 235-244, 2010.

[S38] A. Marchetto, F. Ricca, and P. Tonella, "Empirical Validation of a Web Fault Taxonomy and its usage for Fault Seeding," IEEE International Workshop on Web Site Evolution, pp. 31–38, 2007.

[S39] H. Bajwa, W. Xiong, and F. Maurer, "Evaluating Current Testing Processes of Web-Portal Applications," Web Engineering Springer Berlin Heidelberg, vol. 1, no. 403, pp. 603–605, 2005.

[S40] S. Artzi, A. Kiezun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M. D. Ernst, "Finding bugs in dynamic web applications," Proceedings of the international symposium on Software testing and analysis, p. 261, 2008.

[S41] B. Stepien, L. Peyton, and P. Xiong, "Framework testing of web applications using TTCN-3," International Journal on Software Tools for Technology Transfer, vol. 10, no. 4, pp. 371–381, Apr. 2008.

[S42] E. C. B. Matos and T. C. Sousa, "From formal requirements to automated web testing and prototyping," Innovations in Systems and Software Engineering, vol. 6, no. 1–2, pp. 163–169, Jan. 2010.

[S43] S. Thummalapenta, K. V. Lakshmi, S. Sinha, N. Sinha, and S. Chandra, "Guided Test Generation for Web Applications," Proceedings of the International Conference on Software Engineering, pp. 162–171, 2013.

[S44] H. Raffelt, T. Margaria, B. Steffen, and M. Merten, "Hybrid test of web applications with webtest," Proceedings of workshop on Testing analysis and verification of web services and applications, pp. 1–7, 2008.

[S45] W. G. J. Halfond and A. Orso, "Improving test case generation for web applications using automated interface discovery," Proceedings of the joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, p. 145, 2007.

[S46] N. Alshahwan, M. Harman, a Marchetto, and P. Tonella, "Improving web application testing using testability measures," IEEE International Symposium on Web Systems Evolution, pp. 49–58, 2009.

[S47] S. Sampath, S. Sprenkle, E. Gibson, and L. Pollock, "Integrating Customized Test Requirements with Traditional Requirements in Web Application Testing," Proceedings of the workshop on Testing, analysis, and verification of web services and applications, pp. 23-32, 2006

[S48] A. Mesbah, & A. Van Deursen, "Invariant-based automatic testing of AJAX user interfaces," Proceedings of IEEE International Conference on Software Engineering, pp. 210-220, 2009

[S49] A. Mesbah, A. Van Deursen and D. Roest, "Invariant-Based Automatic Testing of Modern Web Applications," IEEE Transactions on Software Engineering, vol. 38, no. 1, pp. 35–53, 2012.

[S50] S. Mirshokraie and A. Mesbah, "JSART: JavaScript Assertion-based Regression Testing," Web Engineering Springer Berlin Heidelberg, no. 1, pp. 238-252, 2012

[S51] S. Sprenkle, C. Cobb, and L. Pollock, "Leveraging User-Privilege Classification to Customize Usage-based Statistical Models of Web Applications." IEEE International Conference on Software Testing, Verification and Validation, pp. 161-170, 2012.

[S52] S. Elbaum, G. Rothermel, S. Karre, and M. F. Ii, "Leveraging user-session data to support Web application testing," IEEE Transactions on Software Engineering, vol. 31, no. 3, pp. 187–202, 2005.

[S53] B. Bordbar and K. Anastasakis, "MDA and Analysis of Web Applications," Trends in Enterprise Application Architecture, Springer Berlin Heidelberg, pp. 44–55, 2006.

[S54] C. Kallepalli and J. Tian, "Measuring and modeling usage and reliability for statistical Web testing," IEEE Transactions on Software Engineering, vol. 27, no. 11, pp. 1023–1036, 2001.

[S55] P. Koopman, P. Achten, and R. Plasmeijer, "Model-based testing of thin-client web applications and navigation input," Lecture Notes in Computer Science including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 4902 LNCS, pp. 299–315, 2007.

[S56] J. Ernits, R. Roo, J. Jacky, and M. Veanes, "Model-Based Testing of Web Applications Using NModel," Proceedings of the IFIP WG International Conference on Testing of Software and Communication Systems and International FATES Workshop, vol. 5826, pp. 211–216, 2009.

[S57] K. Dobolyi and W. Weimer, "Modeling consumer-perceived web application fault severities for testing," Proceedings of the international symposium on Software testing and analysis, p. 97, 2010.

[S58] J. Offutt and Y. Wu, "Modeling presentation layers of web applications for testing," Software & Systems Modeling, vol. 9, no. 2, pp. 257–280, 2009

[S59] S. H. Jensen and A. Møller, "Modeling the HTML DOM and Browser API in Static Analysis of JavaScript Web Applications." Proceedings of the ACM SIGSOFT symposium and the European conference on Foundations of software engineering, pp. 59-69, 2011

[S60] B. Song and H. Miao, "Modeling Web Applications and Generating Tests: A Combination and Interactions-guided Approach," IEEE International Symposium on Theoretical Aspects of Software Engineering, no. 2007, pp. 174–181, 2009

[S61] N. Li, T. Xie, M. Jin, and C. Liu, "Perturbation-based user-input-validation testing of web applications," Journal of Systems and Software, vol. 83, no. 11, pp. 2263–2274, Nov. 2010.

[S62] K. Sakamoto, K. Tomohiro, D. Hamura, H. Washizaki and Y. Fukazawa, "POGen: a test code generator based on template variable coverage in gray-box integration testing for web applications," Fundamental Approaches to Software Engineering , Springer Berlin Heidelberg, pp. 343-358., 2013

[S63] S. Artzi and M. Pistoia, "Practical Fault Localization for Dynamic Web Applications," Proceedings of the 2nd ACM/IEEE International Conference on Software Engineering, vol 1, pp. 265–274, May 2010

[S64] W. G. J. Halfond, S. Anand, and A. Orso, "Precise interface identification to improve testing and analysis of web applications," Proceedings of the international symposium on Software testing and analysis, p. 285, 2009.

[S65] S. Sampath and C. Bryce, "Prioritizing User-session-based Test Cases for Web Applications Testing," International Conference on Software Testing, Verification, and Validation, no. 1, pp. 141-150, 2008.

[S66] D. Roest, A. Mesbah, and A. Van Deursen, "Regression Testing Ajax Applications: Coping with Dynamism," International Conference on Software Testing, Verification and Validation, pp. 127–136, 2010.

[S67] D. Amalfitano, A. R. Fasolino, and P. Tramontana, "Rich Internet Application Testing Using Execution Trace Data," International Conference on Software Testing, Verification, and Validation Workshops, pp. 274–283, Apr. 2010.

[S68] A. a. Andrews, J. Offutt, C. Dyreson, C. J. Mallery, K. Jerath, and R. Alexander, "Scalability issues with using FSMWeb to test web applications," Information and Software Technology, vol. 52, no. 1, pp. 52–66, Jan. 2010.

[S69] A. Marchetto, P. Tonella, and F. Ricca, "State-Based Testing of Ajax Web Applications," International Conference on Software Testing, Verification, and Validation, pp. 121–130, 2008.

[S70] A. Marchetto, "Talking about a Mutation-Based Reverse Engineering for Web Testing: A Preliminary Experiment," International Conference on Software Engineering Research, Management and Applications, pp. 161–168, 2008.

[S71] Y. Gerlits, "Testing AJAX functionality with UniTESK," Proceedings of the 4th Spring/Summer Young Researchers' Colloquium on Software Engineering, pp. 50–57, 2010.

[S72] Y. Minamide and D. S. Engineering, "Static Approximation of Dynamically Generated Web Pages," Proceedings of international conference on World Wide Web, pp. 432-441, 2005.

[S73] Y. Zheng, T. Bao, X. Zhang, and W. Lafayette, "Statically Locating Web Application Bugs Caused by Asynchronous Calls," Distribution, vol. 195, no. 6, pp. 805–814, 2011.

[S74] P. Tonella and F. Ricca, "Statistical testing of Web applications," Journal of Software Maintenance and Evolution: Research and Practice, vol. 16, no. 12, pp. 103–127, 2004.

[S75] C.-H. L. C.-H. Liu, D. C. Kung, P. H. P. Hsia, and C.-T. H. C.-T. Hsu, "Structural testing of Web applications," Proceedings International Symposium on Software Reliability Engineering ISSRE 2000, pp. 84–96, 2000.

[S76] H. Liu and H. B. Kuan Tan, "Testing input validation in Web applications through automated model recovery," Journal of Systems and Software, vol. 81, no. 2, pp. 222–233, 2008.

[S77] F. Ricca and P. Tonella, "Testing Processes of Web Applications," Annals of software engineering, 14(1-4), pp. 93–114, 2002.

[S78] N. Mansour and M. Houri, "Testing web applications," Information and Software Technology, vol. 48, no. 1, pp. 31–42, Jan. 2006.

[S79] G. D. Lucca, A. Fasolino, and F. Faralli, "Testing web applications," Proceedings of the International Conference on Software Maintenance , pp. 310–319, 2002.

[S80] A. A. Andrews, J. Offutt, and R. T. Alexander, "Testing Web applications by modeling with FSMs," Software Systems Modeling, vol. 4, no. 3, pp. 326–345, 2005.

[S81] C. Bellettini, A. Marchetto, A. Trentini, and V. Comelico, "TestUml : user-metrics driven Web Applications testing," Proceedings of the ACM symposium on Applied computing, pp. 1694–1698, 2005.

[S82] D. Amyot, J. Roy and M. Weiss, "UCM-Driven Testing of Web Applications," Proceedings of the International SDL Forum, pp. 247–264, 2005.

[S83] J. Hao and E. Mendes, "Usage-based statistical testing of web applications," Proceedings of the International Conference on Web Engineering (ICWE), pp. 17–24, 2006.

[S84] X. Peng and L. Lu, "User-session-based automatic test case generation using GA," Journal of Physical Sciences, vol. 6, no. 13, pp. 3232–3245, 2011.

[S85] 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, Dec. 2010.

[S86] D. Licata and S. Krishnamurthi, "Verifying interactive web programs," Proceedings of the International Conference on Automated Software Engineering , pp. 164–173, 2004.

[S87] M. Benedikt, J. Freire, and P. Godefroid, "VeriWeb : Automatically Testing Dynamic Web Sites." Proceedings of the International World Wide Web Conference (WWW), 2002

[S88] D. Castelluccia, M. Mongiello, M. Ruta, and R. Totaro, "WAVer: A Model Checking-based Tool to Verify Web Application Design," Electronic Notes in Theoretical Computer Science, vol. 157, no. 1, pp. 61–76, May 2006.

[S89] P. Tonella and F. Ricca, "Web Application Slicing in Presence of Dynamic Code Generation," Automated Software Engineering, vol. 12, no. 2, pp. 259–288, 2005.

[S90] S. Sampath, S. Sprenkle, E. Gibson, and L. Pollock, "Web Application Testing with Customized Test Requirements - An Experimental Comparison Study," International Symposium on Software Reliability Engineering, pp. 266–278, 2006.

[S91] J. Tian and L. I. Ma, “Web Testing for Reliability Improvement,” Advances in Computers, 67, pp. 178–225, 2006.

[S92] S. R. Choudhary, H. Versee, and a Orso, “WEBDIFF: Automated identification of cross-browser issues in web applications,” Software Maintenance IEEE International Conference on, pp. 1–10, 2010.

[S93] V. Dallmeier, M. Burger, T. Orth, and A. Zeller, “WebMate : Generating Test Cases for Web 2.0,” Software Quality. Increasing Value in Software and Systems Development, pp. 55-69, 2013.

[S94] S. Sprenkle, H. Esquivel, B. Hazelwood, and L. Pollock, “WEBVIZOR: A Visualization Tool for Applying Automated Oracles and Analyzing Test Results of Web Applications,” pp. 89–93, 2008.

[S95] W. Xiong, H. Bajwa, and F. Maurer, “WIT: A Framework for In-container Testing of Web-Portal Applications 2 Problems Needed to Be Addressed by ICT,” vol. 1, no. 403, pp. 87–97, 2005.

Other References

[1] Business Internet Group San Francisco (BIG-SF), "Black Friday Report on Web Application Integrity," *The Business Internet Group of San Francisco, http://www.tealeaf.com/news/press_releases/2003/0203.asp*, 2003.

[2] V. Garousi, A. Mesbah, A. Betin-Can, and S. Mirshokraie, "A Systematic Mapping of Web Application Testing," *Information and Software Technology, in press,* 2013.

[3] B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software engineering," in Evidence-Based Software Engineering," *Evidence-Based Software Engineering,* 2007.

[4] W. Afzal, R. Torkar, and R. Feldt, "A systematic mapping study on non-functional search-based software testing," in *International Conference on Software Engineering and Knowledge Engineering*, 2008, pp. 488-493.

[5] M. Palacios, J. García-Fanjul, and J. Tuya, "Testing in service oriented architectures with dynamic binding: A mapping study," *Information and Software Technology,* vol. 53, pp. 171-189, 2011.

[6] Z. A. Barmi, A. H. Ebrahimi, and R. Feldt, "Alignment of requirements specification and testing: A systematic mapping study," in *Proceedings of the IEEE Fourth International Conference on Software Testing, Verification and ValidationWorkshops*, 2011, pp. 476 - 485.

[7] P. A. d. M. S. Neto, I. d. C. Machado, J. D. McGregord, E. S. d. Almeida, and S. R. d. L. Meira, "A systematic mapping study of software product lines testing," *Information and Software Technology,* vol. 53, pp. 407 - 423, 2011.

[8] E. Engström and P. Runeson, "Software product line testing - a systematic mapping study," *Journal of Information and Software Technology,* vol. 53, pp. 2 - 13, 2011.

[9] C. R. L. Neto, P. A. d. M. S. Neto, E. S. d. Almeida, and S. R. d. L. Meira, "A mapping study on software product lines testing tools," in *Proceedings of International Conference on Software Engineering and Knowledge Engineering*, 2012.

[10] W. Afzal, R. Torkar, and R. Feldt, "A systematic review of search-based testing for non-functional system properties," *Information and Software Technology,* vol. 51, pp. 957 - 976, 2009.

[11] Z. Zakaria, R. Atan, A. A. A. Ghani, and N. F. M. Sani, "Unit testing approaches for BPEL: a systematic review," in *Proceedings of the Asia-Pacific Software Engineering Conference*, 2009, pp. 316 - 322.

[12] A. T. Endo and A. d. S. Simao, "A systematic review on formal testing approaches for web services," in *Brazilian Workshop on Systematic and Automated Software Testing, International Conference on Testing Software and Systems*, 2010, p. 89.

[13] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege, "A systematic review of the application and empirical investigation of searchbased test case generation," *IEEE Transactions on Software Engineering,* vol. 36, pp. 742 - 762, 2010.

[14] E. Engström, P. Runeson, and M. Skoglund, "A systematic review on regression test selection techniques," *Journal of Information and SoftwareTechnology,* vol. 53, pp. 14 - 30, 2010.

[15] R. V. Binder, "Testing object-oriented software: a survey," in *Proceedings of the Tools-23: Technology of Object-Oriented Languages and Systems*, 1996, p. 374.

[16] N. Juristo, A. M. Moreno, and S. Vegas, "Reviewing 25 years of testing technique experiments," *Empirical Software Engineering,* vol. 9, pp. 7 - 44, 2004.

[17] P. McMinn, "Search-based software test data generation: a survey: Research articles," *Software Testing, Verification & Reliability,* vol. 14, pp. 105 - 156, 2004.

[18] M. Grindal, J. Offutt, and S. F. Andler, "Combination testing strategies: A survey," *Software Testing, Verification, and Reliability,* vol. 15, 2005.

[19] G. Canfora and M. D. Penta, "Service-oriented architectures testing: a survey," in *International Summer Schools on Software Engineering*, 2008, pp. 78 –105.

[20] C. S. Păsăreanu and W. Visser, "A survey of new trends in symbolic execution for software testing and analysis," *International Journal on Software Tools for Technology Transfer,* vol. 11, pp. 339 - 353, 2009.

[21] M. Bozkurt, M. Harman, and Y. Hassoun, "Testing web services: a survey," Technical Report TR-10-01, Department of Computer Science, King's College London2010.

[22] Y. Jia and M. Harman, "An analysis and survey of the development of mutation testing," *IEEE Transactions of Software Engineering,* vol. 37, pp. 649 – 678, 2011.

[23] P. A. d. M. S. Neto, P. Runeson, I. d. C. Machado, E. S. d. Almeida, S. R. d. L. Meira, and E. Engström, "Testing Software Product Lines," *IEEE Software,* vol. 28, pp. 16 - 20, 2011.

[24] A. M. Memon and B. N. Nguyen, "Advances in automated model-based system testing of software applications with a GUI front-end," *Advances in Computers,* vol. 80, pp. 121–162, 2010.

[25] T. D. Hellmann, A. Hosseini-Khayat, and F. Maurer, "Agile Interaction Design and Test-Driven Development of User Interfaces - A Literature Review," in *Agile Software Development: Current Research and Future Directions*, 2010.

[26] I. Banerjee, B. Nguyen, V. Garousi, and A. Memon, "Graphical User Interface (GUI) Testing: Systematic Mapping and Repository," *Information and Software Technology, in press,* 2013.

[27] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," presented at the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2008.

[28] Y. Jia and M. Harman, " Mutation Testing Paper Repository," in *http://www.dcs.kcl.ac.uk/pg/jiayue/repository*, Last accessed : July 2013.

[29] Y. Zhang, "Repository of Publications on Search based Software Engineering," in *http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/*, Last accessed : July 2013.

[30] V. Garousi, "Online Paper Repository for Software Test-Code Engineering: A Systematic Mapping," in *http://www.softqual.ucalgary.ca/projects/SM/STCE*, Last accessed : Dec. 5, 2012.

[31] V. Garousi, "Online Paper Repository for GUI Testing," in http://www.softqual.ucalgary.ca/projects/2012/GUI_SM/, Last accessed : Dec. 5, 2012.

[32] V. Garousi, "Online Paper Repository for Developing Scientific Software: A Systematic Mapping," in http://softqual.ucalgary.ca/projects/2012/SM_CSE/, Last accessed : Dec. 1, 2012.

[33] S. Dogan, A. Betin-Can, and V. Garousi, "Online SLR Data for Web Application Testing (WAT)," in http://goo.gl/VayAr, Last accessed : July 15, 2013.

[34] B. Kam and T. R. Dean, "Lessons Learned from a Survey of Web Applications Testing," in *Proceedings of the International Conference on Information Technology: New Generations*, 2009.

[35] M. H. Alalfi, J. R. Cordy, and T. R. Dean, "Modelling methods for web application verification and testing: state of the art," *Journal Software Testing, Verification & Reliability,* vol. 19, pp. 265-296, 2009.

[36] G. A. D. Lucca and A. R. Fasolino, "Testing Web-based applications: The state of the art and future trends," *Information and Software Technology,* vol. 48, pp. 1172-1186, 2006

[37] D. Amalfitano, A. Fasolino, and P. Tramontana, "Techniques and tools for rich internet applications testing," in *Proceedings IEEE International Symposium on Web Systems Evolution*, 2010, pp. 63–72.

[38] E. Insfran and A. Fernandez, "A systematic review of usability evaluation in Web development," in *Proceedings of Web Information Systems Engineering*, 2008, pp. 81-91.

[39] F. Elberzhager, J. Münch, and V. T. N. Nha, "A systematic mapping study on the combination of static and dynamic quality assurance techniques," *Information and Software Technology,* vol. 54, pp. 1-15, 2012.

[40] D. S. Cruzes and T. Dybå, "Synthesizing Evidence in Software Engineering Research," in *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2010

[41] D. S. Cruzes and T. Dybå, "Recommended Steps for Thematic Synthesis in Software Engineering," in *Proc. International Symposium on Empirical Software Engineering and Measurement*, 2011, pp. 275-284.

[42] D. S. Cruzesa and T. Dybåb, "Research synthesis in software engineering: A tertiary study," *Information and Software Technology,* vol. 53, pp. 440–455, 2011.

[43] G. S. Waliaa and J. C. Carverb, "A systematic literature review to identify and classify software requirement errors," *Information and Software Technology,* vol. 51, pp. 1087–1109, 2009.

[44] H. Cooper, L. V. Hedges, and J. C. Valentine, "The Handbook of Research Synthesis and Meta-Analysis," 2nd ed: Russell Sage Foundation, 2009.

[45] S. R. S. Souza, M. A. S. Brito, R. A. Silva, P. S. L. Souza, and E. Zaluska, "Research in Concurrent Software Testing: A Systematic Review," in *Proceedings of the Workshop on Parallel and Distributed Systems: Testing, Analysis, and Debugging*, 2011, pp. 1-5.

[46] J. H. Andrews, L. C. Briand, Y. Labiche, and A. S. Namin, "Using Mutation Analysis for Assessing and Comparing Testing Coverage Criteria," *IEEE Transactions on Software Engineering,* vol. 32, pp. 608-624, 2006.

[47] E.J.Weyuker, "Can we measure software testing effectiveness?," in *Proc. of IEEE Software Metrics Symposium*, 1993, pp. 100-107.

[48] F. Shull, J. Singer, and D. I. K. Sjøberg, *Guide to Advanced Empirical Software Engineering*: Springer, 2008.

[49] R. Feldt and A. Magazinius, "Validity Threats in Empirical Software Engineering Research-An Initial Survey," in *Proceedings of the Software Engineering and Knowledge Engineering Conference*, 2010.

[50] N. K. Liborg, "A study of threats to validity in controlled software engineering experiments," *Masters thesis, University of Oslo, https://www.duo.uio.no/handle/10852/9155*, 2004.

[51] M. Ivarsson and T. Gorschek, "A method for evaluating rigor and industrial relevance of technology evaluations," *Empir Software Eng,* vol. 16, pp. 365–395, 2011.

[52] M. J. Harrold, R. Gupta, and M. L. Soffa, "A methodology for controlling the size of a test suite," *ACM Transactions on Software Engineering and Methodology,* vol. 2, pp. 270 - 285, 1993.

[53] Y. Deng, P. Frankl, and J. Wang, "Testing web database applications," *ACM SIGSOFT Software Engineering Notes,* vol. 29, pp. 1-10, 2004.

[54] S. Eldh, H. Hansson, S. Punnekkat, A. Pettersson, and D. Sundmark, "A framework for comparing efficiency, effectiveness and applicability of software testing techniques," in *Testing: Academic & Industrial Conference-Practice And Research Techniques*, 2006, pp. 159 - 170.

[55] W. G. Halfond, and A. Orso, "Command-form coverage for testing database applications," 21st IEEE/ACM International Conference on Automated Software Engineering ASE'06, pp. 69-80, 2006

TEZ FOTOKOPİ İZİN FORMU

ENSTİTÜ

Fen Bilimleri Enstitüsü	☐
Sosyal Bilimler Enstitüsü	☐
Uygulamalı Matematik Enstitüsü	☐
Enformatik Enstitüsü	☒
Deniz Bilimleri Enstitüsü	☐

YAZARIN

Soyadı	:	Doğan
Adı	:	Serdar
Bölümü	:	Bilişim Sistemleri

TEZİN ADI (İngilizce) : Web Application Testing: A Systematic Literature Review

TEZİN TÜRÜ : Yüksek Lisans ☒ Doktora ☐

1. Tezimin tamamı dünya çapında erişime açılsın ve kaynak gösterilmek şartıyla tezimin bir kısmı veya tamamının fotokopisi alınsın. ☒

2. Tezimin tamamı yalnızca Orta Doğu Teknik Üniversitesi kullancılarının erişimine açılsın. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.) ☐

3. Tezim bir (1) yıl süreyle erişime kapalı olsun. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.) ☐

Yazarın imzası .... Tarih ....