Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Web application testing: A systematic literature review

Serdar Doğan[a,b], Aysu Betin-Can[a], Vahid Garousi[a,c,d,*]

[a] Department of Information Systems, Informatics Institute, Middle East Technical University, Ankara, Turkey
[b] ASELSAN A.S., Ankara, Turkey
[c] Software Quality Engineering Research Group (SoftQual), Department of Electrical and Computer Engineering, Schulich School of Engineering, University of Calgary, Calgary, Alberta, Canada
[d] Department of Software Engineering, Atilim University, Ankara, Turkey

ARTICLE INFO

Article history:
Received 26 August 2013
Received in revised form
16 November 2013
Accepted 5 January 2014
Available online 21 January 2014

Keywords:
Systematic literature review
Web application
Testing

ABSTRACT

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.
Methods: We systematically review the body of knowledge related to functional testing of web application through a systematic literature review (SLR) study. This SLR is a follow-up and complimentary study to a recent systematic mapping (SM) study that we 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, and (4) the state of empirical evidence and industrial relevance.
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.

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) (BIG-SF, 2003) 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

* Corresponding author at: Software Quality Engineering Research Group (SoftQual), Department of Electrical and Computer Engineering, Schulich School of Engineering, University of Calgary, Calgary, Alberta, Canada. Tel.: +1 403 210 5412.
E-mail addresses: serdardogan@aselsan.com.tr, serdar.dogan@metu.edu.tr (S. Doğan), betincan@metu.edu.tr (A. Betin-Can), vgarousi@ucalgary.ca, vahid.garousi@atilim.edu.tr (V. Garousi).

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 paper, we present a systematic literature review (SLR) of the web application testing (WAT) research domain.

In a recent work, we conducted a systematic mapping (SM) study (Garousi et al., 2013) in which we reviewed 79 papers in the WAT domain. The current SLR is a follow-up complementary study after our SM study. We continue in this SLR the secondary study that we started in our SM by focusing in depth into the empirical and evidence-base aspects of the WAT domain. The SM and the SLR studies have been conducted by paying close attention to major differences between these two types of secondary studies, e.g., refer to the guideline by Kitchenham and Charters (2007).

To the best of our knowledge, this paper is the first SLR in the area of WAT. The remainder of this paper 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.

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 (Garousi et al., 2013; Afzal et al., 2008; Palacios et al., 2011; Barmi et al., 2011; Neto et al., 2011a,b; Engström and Runeson, 2011; Neto et al., 2012; Afzal et al., 2009; Zakaria et al., 2009; Endo and Simao, 2010; Ali et al., 2010; Engström et al., 2010; Binder, 1996; Juristo et al., 2004; McMinn, 2004; Grindal et al., 2005; Canfora and Penta, 2008; Păsăreanu and Visser, 2009; Bozkurt et al., 2010; Jia and Harman, 2011; Memon and Nguyen, 2010; Hellmann et al., 2010; Banerjee et al., 2013) 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 (Hellmann et al., 2010) to 264 (Jia and Harman, 2011).

Our recent SM study (Garousi et al., 2013) reviewed 79 papers in the WAT domain and is considered the first step (phase) of the current SLR. The mapping (scoping) that we conducted in the SM study enabled us to classify the papers and identify empirical studies. We continue in this SLR the secondary study that we started in our 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. (2008) and Kitchenham and Charters (2007), the goal and scope of SM and SLR studies are quite different and we follow those distinctions between our previous SM (Garousi et al., 2013) and this SLR.

2.2. Online paper repositories in software engineering

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 (Jia and Harman, 2013), and another in the area of search-based software engineering (SBSE) (Zhang, 2013). In

Table 1
22 Secondary studies in software testing.

Type of secondary study	Secondary study area	Number of primary studies	Year	Reference
SM	Non-functional search-based testing	35	2008	Afzal et al. (2008)
	SOA testing	33	2011	Palacios et al. (2011)
	Testing using requirements specification	35	2011	Barmi et al. (2011)
	Product lines testing	45	2011	Neto et al. (2011a)
	Product lines testing	64	2011	Engström and Runeson (2011)
	Product lines testing tools	33	2012	Neto et al. (2012)
	Web application testing	79	2013	Garousi et al. (2013)
	Graphical User Interface (GUI) testing	136	2013	Banerjee et al. (2013)
SLR	Search-based non-functional testing	35	2009	Afzal et al. (2009)
	Unit testing for Business Process Execution Language (BPEL)	27	2009	Zakaria et al. (2009)
	Formal testing of web services	37	2010	Endo and Simao (2010)
	Search-based test-case generation	68	2010	Ali et al. (2010)
	Regression test selection techniques	27	2010	Engström et al. (2010)
Survey/analysis	Object-oriented testing	140	1996	Binder (1996)
	Testing techniques experiments	36	2004	Juristo et al. (2004)
	Search-based test data generation	73	2004	McMinn (2004)
	Combinatorial testing	30	2005	Grindal et al. (2005)
	SOA testing	64	2008	Mesbah and Prasad (2011)
	Symbolic execution	70	2009	Păsăreanu and Visser (2009)
	Testing web services	86	2010	Bozkurt et al. (2010)
	Mutation testing	264	2011	Jia and Harman (2011)
	Product lines testing	16	2011	Neto et al. (2011b)
Taxonomy	Model-based GUI testing	33	2010	Memon and Nguyen (2010)
Literature review	TDD of user interfaces	6	2010	Hellmann et al. (2010)

three recent SM studies conducted by us and our colleagues, we also shared the paper repositories online (Garousi, 2012a,b,c).

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 (Dogan et al., 2013).

2.3. Secondary studies in web application testing

We were able to find four secondary studies in the area of WAT (Kam and Dean, 2009; Alalfi et al., 2009; Lucca and Fasolino, 2006; Amalfitano et al., 2010b). 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 (2009) 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. (2009) 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. Lucca and Fasolino (2006) 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. (2010b) 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 Insfran and Fernandez (2008) 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.

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 (2007), 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.

After identifying the need for the review, we specify 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 Fig. 1. The process starts with article selection (discussed in detail in Section 3.3). Then, we adapted the classification scheme that we had developed in the SM study (Garousi et al., 2013) 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.

3.2. Goal and research questions

According to the guideline by Kitchenham and Charters (2007), 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 (Petersen et al., 2008). In other words, the results of a SM can be fed into a more rigorous SLR study to support evidence-based

Table 2
Secondary studies in web application testing.

Paper title	Reference	Year	# of primary studies	Summary
Lessons learned from a survey of web applications testing	Kam and Dean (2009)	2009	20	A survey of 20 papers classifying them into six groups: formal, object-oriented, statistical, UML-based, slicing, and user session-based
Modeling methods for web application verification and testing: state of the art	Alalfi et al. (2009)	2009	24	The study surveyed 24 different modeling methods used in web site verification and testing. Based on a short catalog of desirable properties of web applications that require analysis, two different views of the methods were presented: a general categorization by modeling level, and a detailed comparison based on property coverage
Testing web-based applications: the state of the art and future trends	Lucca and Fasolino (2006)	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	Amalfitano et al. (2010b)	2010	29	A classification framework for rich internet application testing and the list of existing web testing tools

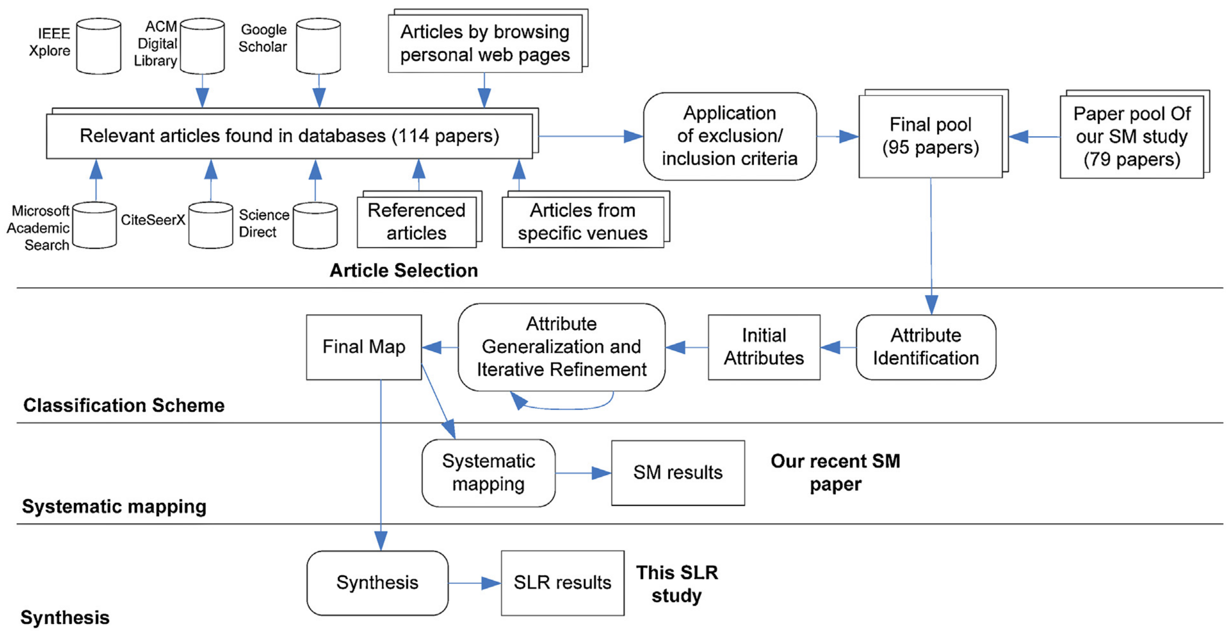


Fig. 1. The review protocol used in this SLR study.

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 formulate 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 (Ali et al., 2010).

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 3 (adapted from Garousi et al., 2013). We can see that the two sets of RQs are different from each other and are complimentary.

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 guidelines (Kitchenham and Charters, 2007; Petersen et al., 2008), to find relevant studies, we searched the following six major online search academic article search engines: (1) IEEE Xplore,[1] (2) ACM Digital Library,[2] (3) Google Scholar,[3] (4) Microsoft Academic Search,[4] (5) CiteSeerX,[5] and (6) Science Direct.[6] These search engines have also been used in other similar studies (e.g., Neto et al., 2011a; Elberzhager et al., 2012).

[1] http://ieeexplore.ieee.org.
[2] http://dl.acm.org.
[3] http://scholar.google.com.
[4] http://academic.research.microsoft.com.
[5] http://citeseerx.ist.psu.edu.
[6] http://www.sciencedirect.com.

Table 3
RQs of the SM study.

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?

(Adapted from Garousi et al., 2013).

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 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 4.

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.

Table 4
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)
```

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 Fig. 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 the 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 these 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.

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 the paper 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 Dogan et al. (2013). The classifications of each selected publication according to the classification scheme presented in Garousi et al. (2013) and also the empirical data extracted from the studies are also available in the online repository.

Fig. 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 to 2013. Note that the since the study was conducted in the midst of the year 2013, thus the data for this year are partial.

3.5. *Research type classification of the articles in the final pool*

The RQ2 and RQ3 in this study focus on empirical studies in WAT, for which we conduct the systematic analysis only on the set of empirical studies in our pool of papers. Therefore, we had to

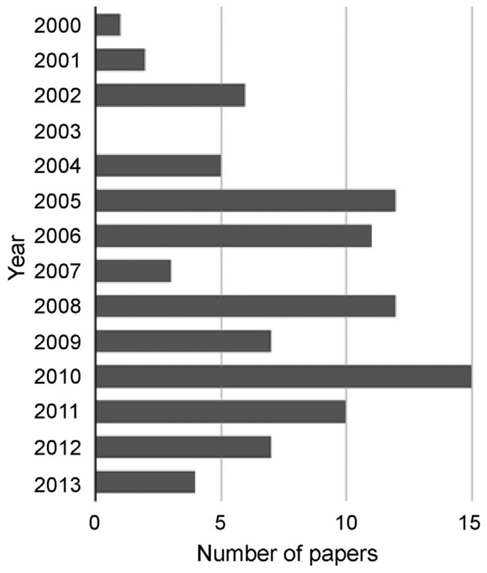


Fig. 2. Annual trend of studies included in our pool.

distinguish the empirical studies in the final pool. We have categorized the 95 primary studies by adopting the classification approach for research facet of papers proposed by Petersen et al. (2008), as follows:

1. Solution proposal: A solution for a problem is proposed, which can be either novel or a significant extension of an existing technique. The potential benefits and the applicability of the solution are shown by a small example or a good line of argumentation.
2. Validation research: Techniques investigated are novel and have not yet been implemented in practice. Techniques used are for example experiments, i.e., work done in the laboratory.
3. Evaluation research: Techniques are implemented in practice and an evaluation of the technique is conducted. That means, it is shown how the technique is implemented in practice (solution implementation) and what are the consequences of the implementation in terms of benefits and drawbacks (implementation evaluation).

In our previous SM study (Garousi et al., 2013), we had conducted such a categorization within the scope of that study, i.e., for the 79 primary studies to map each study based on its research facet according to the approach proposed by Petersen et al. (2008). Here, we provide this categorization for the 95 primary studies in the current pool to show the diversity of the studies and to show which ones have been considered in addressing RQ2 and RQ3. Table 5 shows the list of primary studies in each of these categories. For the RQ2 and RQ3 we examined only the empirical studies which are the articles in the validation and evaluation research category.

3.6. *Data extraction*

As discussed above, our recent SM study was the first step of the current SLR. The mapping (scoping) that we 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 (Dogan et al., 2013).

3.7. *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., Cruzes and Dybå, 2010, 2011a,b), and also other SLRs which had used high-quality synthesis approaches (e.g., Ali et al., 2010; Waliaa and Carverb, 2009).

According to (Cruzes and Dybå, 2010), 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 (Cooper et al., 2009). 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 methods studies, different methods of research synthesis, e.g., thematic analysis and narrative synthesis, are required (Cruzes and Dybå, 2010).

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 (Cruzes and Dybå, 2010, 2011a,b) in this work. We followed the thematic analysis steps recommended by (Cruzes and Dybå, 2011a) 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.

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 Fig. 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

Table 5
Types of the research facet (method) of the articles in our pool (based on the systematic mapping approach followed in our SM study (Garousi et al., 2013) and also the approach proposed by Petersen et al. (2008).

Research facet types	Paper references	Number of papers
Solution proposal	Tonella and Ricca (2002, 2004a,b, 2005), Di Lucca et al. (2006), Törsel (2013), Qi et al. (2006), Ricca and Tonella (2001, 2002a), Alalfi et al. (2010), Offutt et al. (2004), Sampath et al. (2004), Hu (2002), Liu (2006), Di Sciascio et al. (2005), Stepien et al. (2008), Matos and Sousa (2010), Raffelt et al. (2008), Bordbar and Anastasakis (2006), Koopman et al. (2007), Ernits et al. (2009), Offutt and Wu (2009), Song and Miao (2009), Gerlits (2010), Minamide and Engineering (2005), Zheng et al. (2011), Liu et al. (2000), Mansour and Houri (2006), Lucca et al. (2002), Andrews et al. (2005), Bellettini et al. (2005), Amyot et al. (2005), Licata and Krishnamurthi (2004), Benedikt et al. (2002), Castelluccia et al. (2006), and Xiong et al. (2005)	36
Validation research	Artzi et al. (2011), Saxena et al. (2010), Marback et al. (2012), Sampath et al. (2005, 2006a, 2007), Praphamontripong and Offutt (2010), Mesbah and Prasad (2011), Halfond and Orso (2008), Sprenkle et al. (2005b, 2007, 2008, 2012), Harman and Alshahwan (2008), Dobolyi et al. (2010), Ran et al. (2009), Luo et al. (2009), Alshahwan et al. (2012), Choudhary et al. (2012), Ozkinaci and Betin Can (2011), Pattabiraman and Zorn (2010), Ettema and Bunch (2010), Marchetto et al. (2007), Artzi et al. (2008), Thummalapenta et al. (2013), Halfond and Orso (2007), Alshahwan et al. (2009), Mirshokraie and Mesbah (2012), Kallepalli and Tian (2001), Jensen and Møller (2011), Li et al. (2010), Sakamoto et al. (2013), Artzi and Pistoia (2010), Halfond et al. (2009), Amalfitano et al. (2010a), Andrews et al. (2010), Marchetto et al. (2008b), Marchetto (2008), Liu and Kuan Tan (2008), Ricca and Tonella (2002b), Hao and Mendes (2006), Peng and Lu (2011), Tian and Ma (2006), Choudhary et al. (2010), and Dallmeier et al. (2013)	45
Evaluation research	Marchetto et al. (2008a), Sprenkle et al. (2005a, 2011), Alshahwan and Harman (2011), Mirshokraie et al. (2013), Mesbah and Van Deursen (2009), Mesbah et al. (2012), Elbaum et al. (2005), Dobolyi and Weimer (2010), Sampath and Bryce (2008), Roest et al. (2010), Marchetto and Tonella (2010), and Sampath et al. (2006b)	13

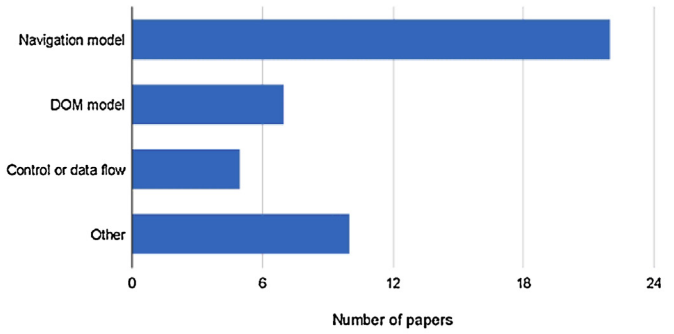
flow models. The “Other” category included models such as: program dependence graphs (PDGs) (Marback et al., 2012), Database Extended Finite State Machine (DEFSM) (Ran et al., 2009), Test architecture as UML models (Hu, 2002), and Unified Markov Models (UMMs) (Kallepalli and Tian, 2001).

We observed that models were used/treated in one of following three cases: (1) as input to the proposed technique, (2) extracted from the web applications and used in subsequent steps of the approach, or (3) as the output of the technique. We classified the primary studies according to this usage purpose of the models and provide the results of this classification in the results showed us that most of the studies are extracting models of WAs in a phase of the technique and use those models as input for consequent phases. Results are shown in Table 6.

For instance, it is worth highlighting the studies (Ricca and Tonella, 2001; Di Sciascio et al., 2005; Bordbar and Anastasakis, 2006) which use the models in different multiple phases of the technique. Ricca and Tonella (2001) and Bordbar and Anastasakis (2006) extract the model of the web applications under test and use this model in their techniques. These two studies also provide the extracted models as output. Previously prepared models of web applications under test are fed as input to techniques proposed in Di Sciascio et al. (2005) and Bordbar and Anastasakis (2006) and these models are transformed to other types of models which are used as input to the consequent steps of the proposed techniques. The results showed us that most of the studies are extracting models of WAs in a phase of the technique and use those models as input for consequent phases.

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 (Souza et al., 2011), we extracted the web fault models and bug taxonomies proposed in the primary studies, as synthesized and summarized in Table 7. Rows of the table are the fault types and the columns are the primary studies discussing or targeting these types of faults. 21 studies discussed fault models and some of them



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]

Fig. 3. Types and frequencies of inferred test models.

Table 6
Classification of the primary studies based on the treatment of test models.

Input to the technique	Törsel (2013), Di Sciascio et al. (2005), Bordbar and Anastasakis (2006) and Ernits et al. (2009)
Extracted and used in the technique	Tonella and Ricca (2004a), Artzi et al. (2011), Marback et al. (2012), Ricca and Tonella (2001), Mesbah and Prasad (2011), Ran et al. (2009), Sampath et al. (2004), Hu (2002), Choudhary et al. (2012), Di Sciascio et al. (2005), Tonella and Ricca (2002), Ettema and Bunch (2010), Thummalapenta et al. (2013), Mesbah and Van Deursen (2009), Mesbah et al. (2012), Sprenkle et al. (2012), Bordbar and Anastasakis (2006), Kallepalli and Tian (2001), Koopman et al. (2007), Roest et al. (2010), Marchetto et al. (2008b), Marchetto (2008), Liu et al. (2000), Liu and Kuan Tan (2008), Mansour and Houri (2006), Lucca et al. (2002), Bellettini et al. (2005), Amyot et al. (2005), Peng and Lu (2011), Marchetto and Tonella (2010), Licata and Krishnamurthi (2004), Castelluccia et al. (2006) and Dallmeier et al. (2013)
Output of the technique	Ricca and Tonella (2001), Liu (2006), Bordbar and Anastasakis (2006), Offutt and Wu (2009) and Song and Miao (2009)

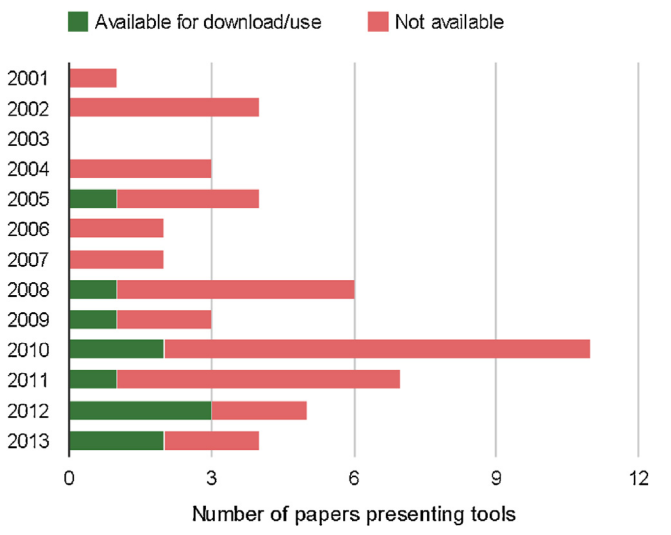


Fig. 4. Number of tools presented in the papers over the years.

conducted fault feeding (mutation testing) based on the proposed types of faults.

It is worth highlighting two of the studies (Mirshokraie et al., 2013; Marchetto et al., 2007) in this context. In Mirshokraie et al. (2013), a bug severity taxonomy was proposed which was used to assess the effectiveness and performance of a mutation testing tool for JavaScript. In Marchetto et al. (2007), 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. Four studies (Praphamontripong and Offutt, 2010; Pattabiraman and Zorn, 2010; Ettema and Bunch, 2010; Marchetto et al., 2008b) proposed more in-depth levels of fault taxonomies. Praphamontripong and Offutt (2010) and Ettema and Bunch (2010) proposed taxonomy for navigation faults and Pattabiraman and Zorn (2010) and Marchetto et al. (2008b) classified asynchronous communication faults, specific to Ajax applications. The raw (before-synthesis) data for the list of fault models/bug taxonomies in each study are provided in Appendix A (Table 27).

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. 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 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, Fig. 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.

Table 7
Fault models/bug taxonomy related to web applications.

Fault models/bug taxonomy	Paper references
Authentication, permission	Marchetto et al. (2008a), Luo et al. (2009), Kallepalli and Tian (2001) and Dobolyi and Weimer (2010)
Internationalization	Marchetto et al. (2008a) and Dobolyi and Weimer (2010)
Functionality	Marchetto et al. (2008a), Sprenkle et al. (2007), Dobolyi and Weimer (2010) and Choudhary et al. (2010)
Portability	Marchetto et al. (2008a)
Navigation	Marchetto et al. (2008a), Praphamontripong and Offutt (2010), Sprenkle et al. (2005b), Luo et al. (2009), Ettema and Bunch (2010), Kallepalli and Tian (2001), Dobolyi and Weimer (2010) and Peng and Lu (2011)
Asynchronous communication	Marchetto et al. (2008a,b) and Pattabiraman and Zorn (2010)
Session	Marchetto et al. (2008a), Marchetto et al. (2007) and Dobolyi and Weimer (2010)
Form construction	Marchetto et al. (2008a), Sprenkle et al. (2007) and Elbaum et al. (2005)
Database, persistance	Sprenkle et al. (2007), Marchetto et al. (2007), Elbaum et al. (2005), Dobolyi and Weimer (2010) and Peng and Lu (2011)
Appearance (GUI, layout, etc.)	Sprenkle et al. (2007), Luo et al. (2009), Dobolyi and Weimer (2010), Peng and Lu (2011) and Choudhary et al. (2010)
Multi tier architectural	Luo et al. (2009)
Syntactic, programming language	Ozkinaci and Betin Can (2011), Artzi et al. (2008) and Artzi and Pistoia (2010)
Cookie manipulation	Marchetto et al. (2007)
Plugin	Marchetto et al. (2007)
Cross-browser incompatibility	Marchetto et al. (2007) and Mesbah et al. (2012)
DOM	Mesbah et al. (2012), Mirshokraie and Mesbah (2012) and Marchetto et al. (2008b)
Scripting	Elbaum et al. (2005) and Jensen and Møller (2011)
Code disclosure	Dobolyi and Weimer (2010)
CSS	Dobolyi and Weimer (2010)

Table 8
Features of the tools available for download/use.

Tool name	Study	Supported testing activity/task	URL	Supported web technology	Comments
MBT4Web	Törsel (2013)	Model-based testing	www.mbt4web.fh-stralsund.de	Generates tool-specific command scripts using web application model in a DSL	Model-based testing framework for web applications
MUTANDIS	Mirshokraie et al. (2013)	Mutation testing	www.github.com/saltlab/mutandis	JavaScript, DOM. Can be applied to any programming language by customization	JavaScript mutation testing tool that leverages static and dynamic program analysis to guide the mutation generation process toward parts of the code that are error-prone or likely to influence the program's output
ATUSA	Mesbah and Van Deursen (2009) and Roest et al. (2010)	UI testing for AJAX	www.crawljax.com	Ajax, DOM	Dependent on Crawljax. Automatically testing UI states of AJAX
Crawljax	Roest et al. (2010)	Reverse engineering	www.crawljax.com	Ajax, DOM	Crawling Ajax-based web applications and reverse engineering of their FSMs
JSART	Mirshokraie and Mesbah (2012)	Regression testing	www.salt.ece.ubc.ca/content/jsart	JavaScript, DOM	Regression testing of JavaScript code based on assertions
Tool-suite: (CreRIA, CrawlRIA, Test Case Generator, Test Case Reducer, DynaRIA)	Amalfitano et al. (2010a)	Regression testing	http://wpage.unina.it/ptramont/downloads.htm	Ajax, DOM	A tool-suite for dynamic analysis and automatic regression testing of rich internet applications
reAJAX	Marchetto and Tonella (2010)	Reverse engineering	selab.fbk.eu/marchetto/tools/ajax/testing1/experimentData.zip	Ajax, DOM	A tool for extracting the FSM of Ajax applications through dynamic and static code analysis
WebMate	Dallmeier et al. (2013)	Reverse engineering Model-based testing Cross-browser compatibility testing	www.degesso.de	HTML, JavaScript, CSS	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
WebVizOr	Sprenkle et al. (2008)	Program comprehension Test oracles	www.eecis.udel.edu/~hiper/webvizor	All types of web applications producing HTML output	A visualization tool for applying automated oracles and analyzing test results of web applications
Web Portal In-container Testing (WIT)	Xiong et al. (2005)	In-container testing (testing in application servers)	sourceforge.net/projects/wit-ict/files/wit-ict	JSR-000168 compliant Java portlets	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

Table 9
Frequencies of cost measures across empirical studies.

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

To further answer RQ1.3, we extracted the features and capabilities of the tools available for download. They are reported in Table 8.

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 (Dogan et al., 2013).

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 9 and discuss them next.

4.2.1.1.1. 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, Marchetto et al. (2008a) 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.).

Saxena et al. (2010), 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. Sprenkle et al. (2005a) 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. Mesbah and Prasad (2011) measured and reported the crawling time (in minutes) needed for the tool to run. Sprenkle et al. (2007) proposed automated oracles for web applications and compared their execution times on four case study SUTs. In the case-study of Sprenkle et al. (2007), the slowest oracle executed in 14 min on average, whereas replaying the test suite took about 90 min.

Mesbah and Van Deursen (2009) 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., Mirshokraie and Mesbah, 2012) 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. Choudhary et al. (2010) 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 5 min to complete.

4.2.1.1.2. Test-suite size. 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 Artzi et al. (2011), 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 (Sprenkle et al., 2005a; Sampath et al., 2006a, 2007; Amalfitano et al., 2010a; Marchetto et al., 2008b; Tonella and Ricca, 2005) had the above-mentioned objective. For instance, Sprenkle et al. (2005a) is an empirical comparison of three test-suite reduction techniques for user-session-based testing. Sampath et al. (2007) examined the impact of three test-suite reduction techniques on cost/benefit of test suites.

4.2.1.1.3. Memory space. For test techniques to be practical, they should have reasonable memory space requirements and should scale up for large SUTs. Seven studies (Sprenkle et al., 2005a,b, 2007, 2012; Harman and Alshahwan, 2008; Sampath et al., 2005, 2006b) measured this metric. We discuss Sprenkle et al. (2005b, 2007) as two representative studies.

Sprenkle et al. (2007) proposed automated oracles for web apps and compared their memory space requirements on four SUTs. Sprenkle et al. (2005b) 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).

4.2.1.1.4. Other. Three studies (Marchetto et al., 2008a; Sprenkle et al., 2011; Andrews et al., 2010) used/proposed other types of cost metrics, which were related to cost and complexity. Marchetto et al. (2008a) measured test-suite complexity which was defined as the number of steps (test commands) required to execute the whole suite. Note that this metric is different than test-suite size. Sprenkle et al. (2011) measured number of states and number of edges in the navigation model, a specific test model that was generated for the purpose of testing. Andrews et al. (2010) 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 10. Detecting injected faults (mutation score) was the most popular metric (used in 25 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.

Code coverage metrics seem to be quite popular in assessing effectiveness of WAT techniques. 23 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 11. Statement (also called line or node) coverage was the most widely used. Both data-flow and control-flow metrics have been used.

Table 10
Frequencies of effectiveness measures across empirical studies.

	Coverage (code)	Coverage (model-based requirements)	Coverage (other)	Detecting real faults	Detecting injected faults	Other	None
Empirical studies ($N = 58$)	28	9	15	17	27	10	9
	29%	9%	16%	18%	28%	11%	15%

Table 11
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%

Table 12
Model coverage metrics used for the purpose of effectiveness measurement.

Model coverage metrics	Studies
State coverage	Thummalapenta et al. (2013), Mesbah et al. (2012), Amalfitano et al. (2010a) and Marchetto et al. (2008b)
Edge (transition) coverage	Sprenkle et al. (2011), Thummalapenta et al. (2013), Amalfitano et al. (2010a), Marchetto et al. (2008b) and Marchetto and Tonella (2010)
Path coverage	Thummalapenta et al. (2013)
Prime path coverage	Sakamoto et al. (2013)

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 Sakamoto et al., 2013). Table 12 shows the references.

15 empirical studies presented and utilized coverage metrics which were not code or model-based. We categorized them under the "Coverage (other)" category in Table 10. We have summarized the list and brief definition of those metrics in Table 13.

10 studies used metrics other than coverage or mutation score for the purpose of effectiveness measurement. Table 14 lists those studies and the names of the metrics. Each metric is briefly discussed next.

As shown in Table 14, 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.

Halfond and Orso (2007) 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. Mesbah et al. (2012) presented an invariant-based

Table 13
Other coverage metrics.

Coverage metrics	Studies
Use-case coverage: number of use cases exercised by a test suite.	Marchetto et al. (2008a)
Event space coverage: number of events in the GUI space of the SUT exercised by a test suite.	Saxena et al. (2010)
All-URL coverage: covering each URL of the application at least once.	Sprenkle et al. (2005a)
Page coverage: every page in the SUT is visited at least once in some test case.	Ricca and Tonella (2001) and
Hyperlink coverage: every hyperlink from every page in the site is traversed at least once.	Ricca and Tonella (2002b)
Three database-specific coverage criteria:	Alalfi et al. (2010)
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.	
Single URLs: requires all URLs in the base URL set of a SUT to be covered at least once by the test suite.	Sampath et al. (2005)
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.	
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 ˆexplored(card_type_record_page.insert)`	Thummalapenta et al. (2013)
Input-parameter coverage: covering all possibilities of input parameters using black-box approaches, e.g., equivalence classing	Halfond and Orso (2007)
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 Halfond and Orso (2006).	Halfond and Orso (2007) and Halfond et al. (2009)
Template variable coverage: covering all the template variable in HTML pages.	Sakamoto et al. (2013)
Input validation coverage (IVC): at least one path in the program's CFG w.r.t. the validation of inputs has been covered.	Liu and Kuan Tan (2008)
All-hyperlinks testing	Mansour and Houri (2006)
All-input-GUI testing	
All-events testing	
Pages, index pages, web object with high coupling	Bellettini et al. (2005)
Request coverage	Peng and Lu (2011)
Data dependence transition relation coverage	
Link dependence transition relation coverage	

Table 14
Other metrics used/proposed for the purpose of effectiveness measurement.

Metrics	Studies
False positives, false negatives	Dobolyi et al. (2010), Choudhary et al. (2012), Mirshokraie and Mesbah (2012), Dobolyi and Weimer (2010) and Roest et al. (2010)
Precision, recall	Dobolyi et al. (2010), Mirshokraie and Mesbah (2012), Dobolyi and Weimer (2010) and Halfond et al. (2009)
Number of interfaces discovered	Halfond and Orso (2007)
Number of DOM violations	Mesbah et al. (2012)
Reliability growth	Kallepalli and Tian (2001)
Test path reduction rates	Marback et al. (2012)

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 Kallepalli and Tian (2001), 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 a software reliability growth models (SRGMs), i.e., Goel-Okumoto model.

Marback et al. (2012) 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.

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 Fig. 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.

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 Fig. 6.

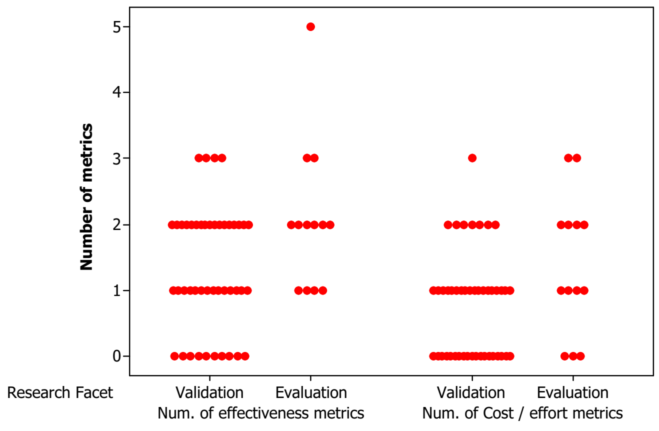


Fig. 5. Usage of multiple metrics in each of the studies, as an individual-value plot.

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., Andrews et al., 2006; Weyuker, 1993) and it thus seems that researchers in the WAT domain have adopted those metrics as well.

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

Identification and discussion of validity threats is one of the most important aspects of empirical research in software engineering (Shull et al., 2008). 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 (Feldt and Magazinius, 2010; Liborg, 2004).

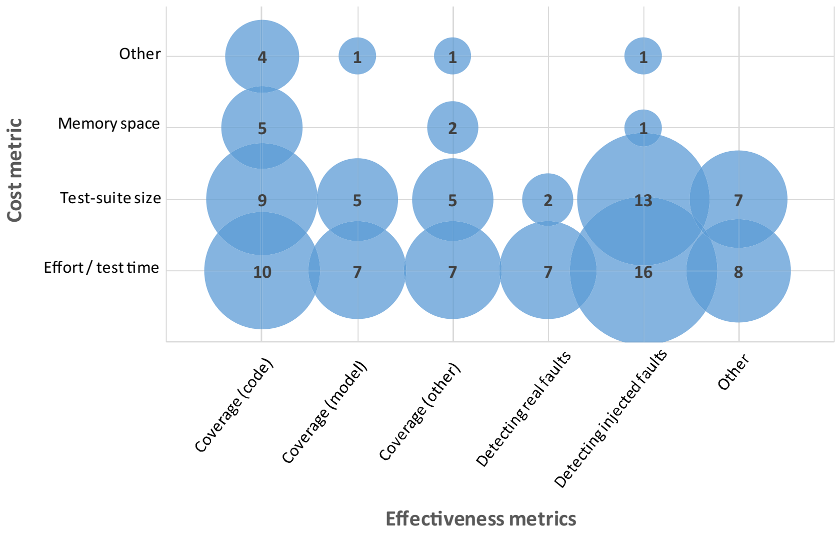


Fig. 6. Bubble-chart of pair of cost and effectiveness metrics used in the studies.

Table 15
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

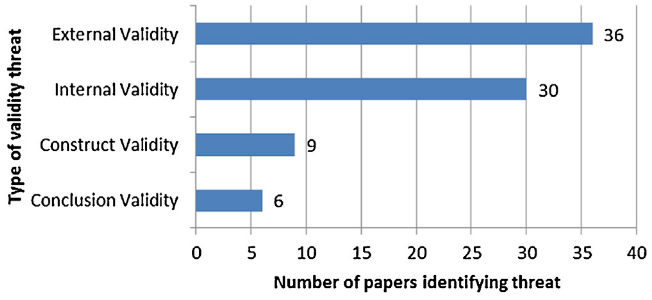


Fig. 7. Number of papers identifying threats in each type.

In our set of primary studies, most of the empirical studies mentioned these threats explicitly, while some have referred to these threats as limitations (Ozkinaci and Betin Can, 2011; Ettema and Bunch, 2010; Sakamoto et al., 2013; Tian and Ma, 2006; Choudhary et al., 2010; Dallmeier et al., 2013). 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 (Shull et al., 2008):

- 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 study. 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.

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 15).

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 (Sprenkle et al., 2005b; Sampath et al., 2005; Sampath and Bryce, 2008), 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 Fig. 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.

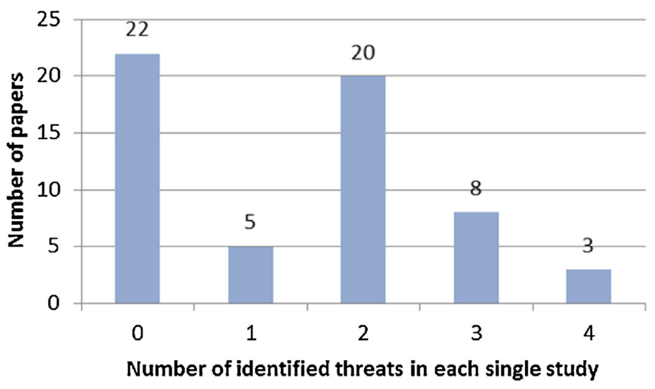


Fig. 8. Number of identified threats in each single study.

We also wanted to assess the number of identified threat types in each single study. Fig. 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 (Marchetto et al., 2007, 2008a; Alshahwan et al., 2012) identified all four types.

We list in Table 16 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.

External validity threats are one of the mostly addressed types of threats. We list in Table 17 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.

We list in Table 18 the list of construct validity threats identified in the empirical studies. 10 distinct types of construct validity threats were identified in 9 papers.

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 these kinds of validity threats are the least addressed threats in the studies. We list in Table 19 the list of conclusion validity threats identified in the empirical studies. 7 distinct types of conclusion validity threats were identified in 6 papers.

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

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

Ivarsson and Gorschek (2011) presented 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.

Table 16
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)	Marchetto et al. (2008a), Praphamontripong and Offutt (2010), Sprenkle et al. (2007), Sprenkle et al. (2005b), Dobolyi et al. (2010), Sampath et al. (2005), Pattabiraman and Zorn (2010), Artzi et al. (2008), Sampath et al. (2006a), Mesbah and Van Deursen (2009), Mesbah et al. (2012), Elbaum et al. (2005), Dobolyi and Weimer (2010), Artzi and Pistoia (2010), Sampath and Bryce (2008), Marchetto and Tonella (2010) and Sampath et al. (2006b)	17
	• Hand-seeded faults		
	• Type of mutations		
	• Faults may not represent real world situations		
	• Even distribution of mutations		
Dependence on 3rd party tools/libraries	• Faults in used 3rd party tools/libraries	Mesbah and Prasad (2011), Halfond and Orso (2007), Mesbah and Van Deursen (2009) and Li et al. (2010)	4
	• Change in implementation of 3rd party tools/libraries may affect results		
Error in human-based identification	• Some steps and identifications done by human	Dobolyi et al. (2010), Alshahwan et al. (2012), Mirshokraie et al. (2013) and Dobolyi and Weimer (2010)	4
	• Human training bias also included in this category		
Simplicity of the SUTs	• More complex SUTs may change the empirical results	Sprenkle et al. (2005a,b)	2
Subjectivity in applying techniques	• Proposed technique includes human dependent manual steps	Marchetto et al. (2008a)	1
Not considering all important metrics	• e.g., cost effectiveness	Elbaum et al. (2005)	1
Representativeness of human subjects	• The experience level and expertise of human subjects in manual steps of the study	Dobolyi and Weimer (2010)	1
Faults in implemented tool	• Quality of the proposed tool	Li et al. (2010)	1
Different behavior of browsers	• Study applied on one or limited number of browsers, results may change on other browsers.	Sprenkle et al. (2007)	1
Limited source of information to derive test artifacts	• e.g., lack of a large number of user sessions that constitute the requirements universe.	Sampath et al. (2005)	1
Small number of classifiers	• In taxonomy studies, having more human classifiers would lead to better results	Marchetto et al. (2007)	1
Automatically generated inputs	• Automatically generated or classified information to generate test artifacts	Sprenkle et al. (2012)	1
Time costs depending on case	• Execution time depends on the machine used or test case characteristics	Sampath and Bryce (2008)	1
Error in oracle comparators	• Oracle comparators can have false positives and false negatives	Sampath and Bryce (2008)	1
Test suite size	• Size of test suite affecting the results	Marchetto and Tonella (2010)	1
Biased SUT selection	• Selection process for the subjects is not randomized. Researcher's bias affects the choice	Dallmeier et al. (2013)	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	Marchetto (2008)	1
Initial results affecting subsequent ones	• e.g., failure in a test case affects next one	Roest et al. (2010)	1
Dependence to database state and configuration	• Behavior of SUTs which are using a database may change according to different state and configuration.	Alshahwan and Harman (2011)	1
Error in human based code modification	• Manually changing statements in a program during a phase of the approach	Marback et al. (2012)	1
	• Removing or fixing any dynamic environment variables manually		
Total number of papers identifying an internal validity threat			30

The model presented in Ivarsson and Gorschek (2011) 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 Ivarsson and Gorschek (2011) are shown in Table 20. We made a slight adjustment to the "scale" aspect: if the sum of

Table 17
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.	Marchetto et al. (2008a), Artzi et al. (2011), Marback et al. (2012), Sampath et al. (2005, 2006a,b, 2007), Praphamontripong and Offutt (2010), Mesbah and Prasad (2011), Sprenkle et al. (2005b, 2007, 2012), Harman and Alshahwan (2008), Alshahwan and Harman (2011), Dobolyi et al. (2010), Alshahwan et al. (2012), Pattabiraman and Zorn (2010), Mirshokraie et al. (2013), Halfond and Orso (2007), Mesbah and Van Deursen (2009), Elbaum et al. (2005), Dobolyi and Weimer (2010), Jensen and Møller (2011), Li et al. (2010), Artzi and Pistoia (2010), Sampath and Bryce (2008), Roest et al. (2010), Marchetto et al. (2008b), Marchetto (2008), Marchetto and Tonella (2010) and Dallmeier et al. (2013)	32
Need for more real-world studies	Empirical comparison needs to be evaluated further by experimenting with more real-world web applications	Sprenkle et al. (2005a, 2011) and Mesbah et al. (2012)	3
Lack of comparison to other studies	Not comparing the study with related studies.	Mirshokraie et al. (2013) and Artzi et al. (2008)	2
Scalability not sure	Empirical study not applied on realistic large scale subjects	Marchetto et al. (2008a)	1
Representativeness of bug dataset	(Mostly in taxonomy studies) The input data set is relatively small.	Marchetto et al. (2007)	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.	Elbaum et al. (2005)	1
Total number of papers identifying an external validity threat			36

Table 18
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	Sampath et al. (2006b)	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.	Halfond and Orso (2007)	1
Subjectivity in bug classification	Human based classification of faults	Marchetto et al. (2007)	1
Inadequate bug location report	Fault localization is weak	Artzi et al. (2008)	1
Not considering the severity of the faults	E.g., potential impact of fault severity on the used techniques and oracle comparators.	Sprenkle et al. (2005b)	1
Subjectivity in choosing metrics	Human based metric selection may include a level of subjectivity	Marchetto et al. (2008a)	1
Faults in implemented tool	Depends on the quality and the reliability of implemented tool	Praphamontripong and Offutt (2010)	1
Dependence on crawler's capabilities,	Quality of 3rd party crawler tool affects would affect the results.	Alshahwan et al. (2012)	1
Dependence on definition of metrics	E.g., different definitions of crawlability metrics may lead to different results	Alshahwan et al. (2012)	1
Dependence on third party tool	E.g., dependence to a third party tool for measuring coverage	Dallmeier et al. (2013)	1
Total number of papers identifying a construct validity threat			9

Table 19
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.	Marchetto (2008)	1
Results rely on interpretation of metrics	E.g., taxonomy assessed through metrics manually.	Marchetto et al. (2007)	1
Results rely on human based classification	E.g., human based classification of faults in taxonomy studies.	Marchetto et al. (2007)	1
Not using statistical tests	E.g., To reject the null hypotheses	Marchetto et al. (2008a)	1
Not including fault severity	conclusions of the experiment could be different if the results were weighted by fault severity	Sampath et al. (2007)	1
Dependence on used statistical technique	Using different statistical techniques may affect the outcome	Alshahwan et al. (2012)	1
Need to maintain the state of the application	Applying same methods and techniques on different state of same SUT may provide different results	Sprenkle et al. (2005a)	1
Total number of papers identifying a conclusion validity threat			6

Table 20
Scoring rubric for evaluating relevance.

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., real-world case study	The research method mentioned to be used in the evaluation does not lend itself to investigate real situations, e.g., conceptual analysis laboratory experiment

Adapted from Ivarsson and Gorschek (2011).

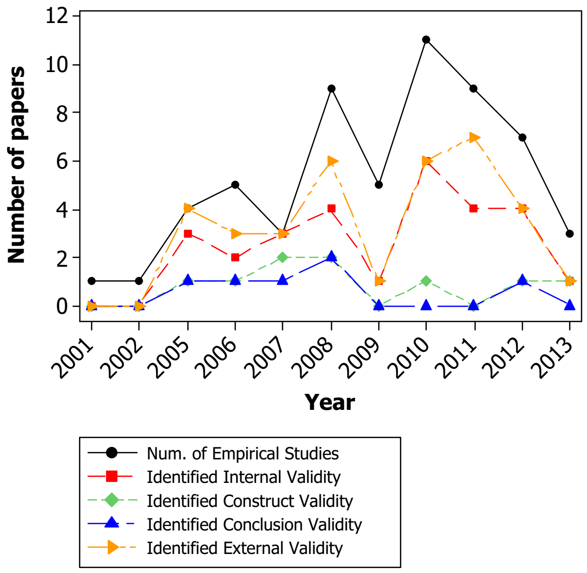


Fig. 9. Trend of the number of empirical studies and type of identified threats over the years.

LOC of the SUTs in a study was less than 1000 LOC, we assigned 0, if it was between 1000 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.

The histograms of Fig. 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 (Ozkinaci and Betin Can, 2011; Ettema and Bunch, 2010; Sakamoto et al., 2013; Tian and Ma, 2006; Choudhary et al., 2010; Dallmeier et al., 2013), and 22 have explained in enough detail.

For 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., real world case study.
- Users/subjects: for 55 papers, the aspect of users was N/A. For the remaining three studies (Harman and Alshahwan, 2008; Marchetto et al., 2007; Tian and Ma, 2006), industry users were involved. In Harman and Alshahwan (2008), a group of 14 participants exercised the SUTs and session data were then collected. In Marchetto et al. (2007), expert testers were hired to classify defects. In Tian and Ma (2006), industry professionals helped in gathering data.

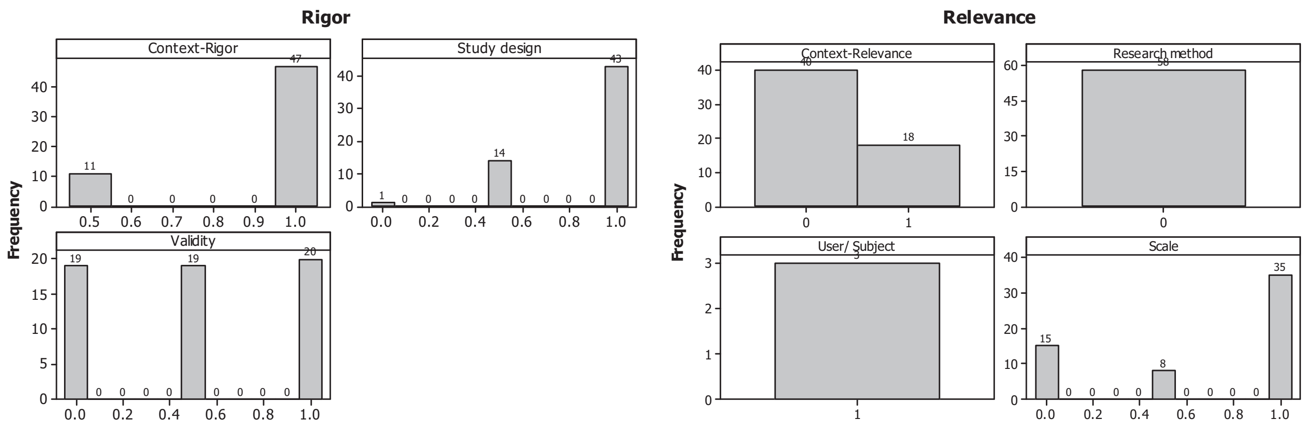
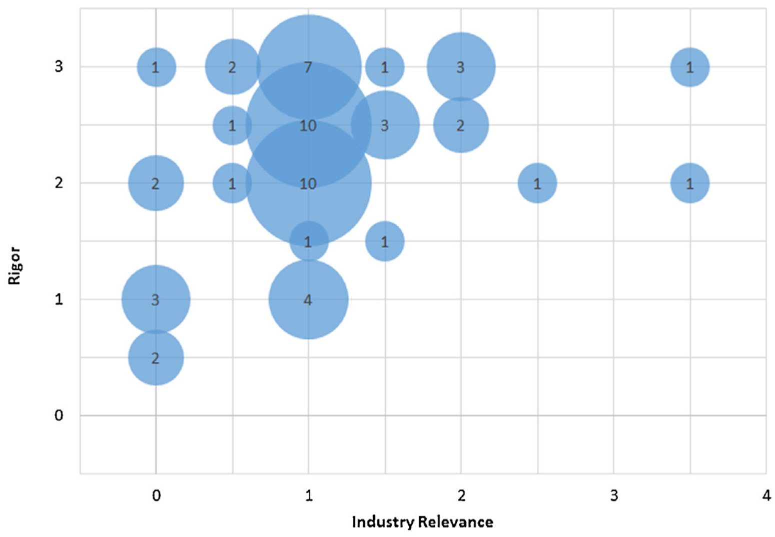


Fig. 10. Histograms of Rigor and Relevance for the 58 empirical studies.



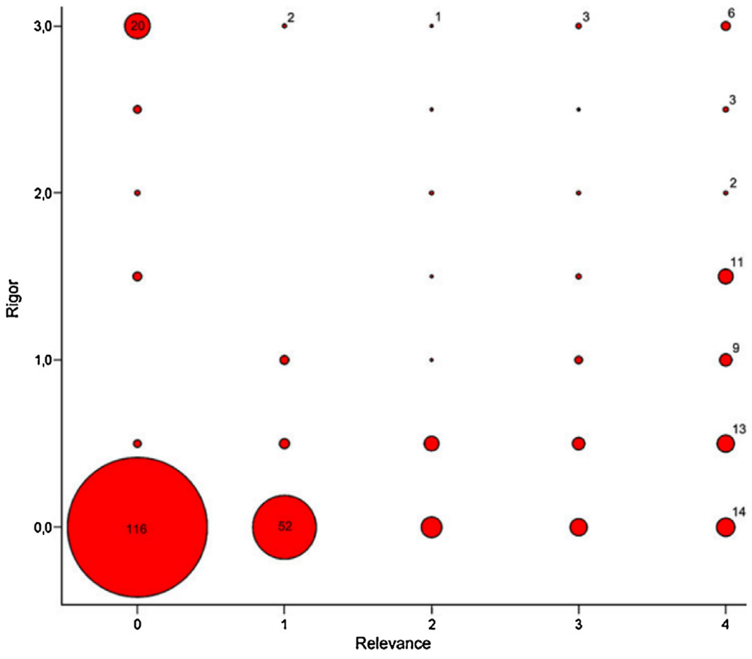


Fig. 11. Rigor versus relevance of the empirical studies in this SLR versus Ivarsson and Gorschek (2011).

- 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.

Our next analysis was to assess the pair of rigor and relevance assessments for each pair, similar to what was done in Ivarsson and Gorschek (2011). The model was applied in Ivarsson and Gorschek (2011) 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 Ivarsson and Gorschek (2011) is shown in Fig. 11. A large portion of studies in Ivarsson and Gorschek (2011), 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 are were of 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 Ivarsson and Gorschek (2011) 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., Shull et al., 2008), it is evident that raising meaningful RQs for an empirical study will help better direct the study and the relevant measurements. Fig. 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.

By putting the results of RQs 2.2 and 2.3 together, similar to other SLRs (e.g., the one on search-based testing (Ali et al., 2010)), 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

Table 21
Studies which empirically analyzed scalability issues.

Study	Approach	Method to study scalability	Summary of results
Harman and Alshahwan (2008)	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 and 40).	The tool was able to complete its entire white-box analysis phase within 3 min for all of the applications considered
Mesbah et al. (2012)	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 min 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
Andrews et al. (2010)	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

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 and industrial relevance 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 (Harman and Alshahwan, 2008; Mesbah et al., 2012; Andrews et al., 2010) explicitly studied scalability and reported the corresponding evidence. A summary of these works are shown in Table 21.

The study reported in Harman and Alshahwan (2008) 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 Harman and Alshahwan (2008) 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 3 min 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 and 150).

The study reported in Mesbah et al. (2012) 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 21.

The study reported in Andrews et al. (2010) 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 21), thus helping the approach become more scalable.

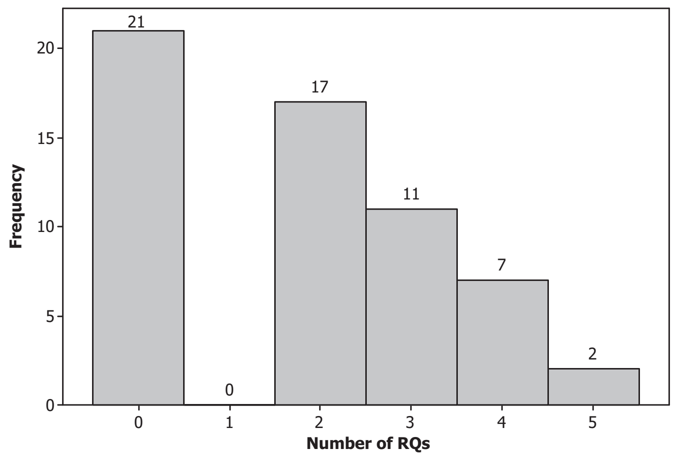


Fig. 12. Number of RQs in empirical studies.

4.3.2. RQ 3.2 – Have different techniques been empirically compared with one another?

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 (Cruzes and Dybå, 2010, 2011a,b) and developed a set of ten WAT theme areas, as shown in Table 22. Frequency of studies under each WAT theme area and the trend of the number of studies under each area are shown in Table 22 and Fig. 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.

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 23). The studies have been divided into the WAT theme areas and we synthesize next the empirical comparisons under each theme area.

Table 22
Frequency of empirical studies under each WAT theme area.

Theme area	# of papers	Percentage (%)	References
White box	7	12	Alshahwan and Harman (2011), Sampath et al. (2005, 2006a), Ozkinaci and Betin Can (2011), Artzi et al. (2008), Halfond and Orso (2007) and Sakamoto et al. (2013)
Black box, FSM, model-based	5	9	Artzi et al. (2011), Sampath et al. (2006a,b), Andrews et al. (2010) and Ricca and Tonella (2002b)
Mutation	3	5	Praphamontripong and Offutt (2010), Mirshokraie et al. (2013) and Marchetto (2008)
AJAX testing	6	10	Marchetto et al. (2008a), Mesbah and Van Deursen (2009), Mesbah et al. (2012), Roest et al. (2010), Marchetto et al. (2008b) and Marchetto and Tonella (2010)
Session-based testing (navigation models, logs)	16	28	Sprenkle et al. (2005a,b, 2011, 2012), Sampath et al. (2007), Harman and Alshahwan (2008), Luo et al. (2009), Ettema and Bunch (2010), Thummalapenta et al. (2013), Elbaum et al. (2005), Kallepalli and Tian (2001), Sampath and Bryce (2008), Amalfitano et al. (2010a), Hao and Mendes (2006), Peng and Lu (2011) and Dallmeier et al. (2013)
Cross-browser compatibility testing	3	5	Mesbah and Prasad (2011) and Choudhary et al. (2010, 2012)
Regression testing	5	9	Marback et al. (2012), Harman and Alshahwan (2008), Dobolyi et al. (2010), Mirshokraie and Mesbah (2012) and Roest et al. (2010)
Oracle	4	7	Sprenkle et al. (2005b, 2007), Ran et al. (2009) and Xiong et al. (2005)
Support for testing	9	16	Saxena et al. (2010), Alshahwan et al. (2012), Marchetto et al. (2007), Alshahwan et al. (2009), Dobolyi and Weimer (2010), Jensen and Møller (2011), Li et al. (2010), Halfond et al. (2009) and Liu and Kuan Tan (2008)
Other	4	7	Halfond and Orso (2008), Pattabiraman and Zorn (2010), Artzi and Pistoia (2010) and Tian and Ma (2006)

4.3.2.1. White-box testing. There was one paper (Artzi et al., 2008), which conducted comparative empirical study in the theme area of white-box testing. The authors of Artzi et al. (2008) 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 Halfond and Orso (2007) for JavaScript testing (referred to as Randomized). Second, they compared their results to those reported by a static analysis technique (Minamide and Engineering, 2005), on the same subject programs.

Apollo test generation strategy outperformed the randomized testing (proposed by Halfond and Orso, 2007) 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 fault for Randomized.

For the three overlapping subject programs, Apollo was both more effective and more efficient than the tool presented in Minamide and Engineering (2005). Apollo found 2.7 times as many HTML validation faults found by Minamide and Engineering (2005)'s tool (120 vs. 45). Apollo found 83 execution faults, which are out of reach for Minamide and Engineering (2005)'s tool. Apollo is also more scalable—on schoolmate, the largest of the programs, Apollo found 104 faults in 10 min, while Minamide and Engineering (2005)'s tool found only 14 faults in 126 min. The authors discussed

Table 23
List of studies conducting empirical comparisons with other studies/tools.

Study	Year of publication	Compared with studies/tools by other authors	Compared with previous studies by the same authors
White-box testing			
Artzi et al. (2008)	2008	Halfond and Orso (2007) and Minamide and Engineering (2005)	–
Session-based testing			
Sprenkle et al. (2005a)	2005	Harrold et al. (1993)	Sampath et al., 2007)
Luo et al. (2009)	2009	Sampath et al. (2007) and Harrold et al. (1993)	–
Sampath et al. (2006a)	2006	Harrold et al. (1993)	–
Elbaum et al. (2005)	2005	Ricca and Tonella (2001)	–
Sampath and Bryce (2008)	2008	Sampath et al. (2007) and Sprenkle et al. (2005b, 2007)	Sampath et al. (2007) and Sprenkle et al. (2005b) (one author in common)
Hao and Mendes (2006)	2006	Kallepalli and Tian (2001)	–
Dallmeier et al. (2013)	2013	A web scraping/crawling framework called Scrappy (www.scrapy.org)	–
AJAX testing			
Marchetto et al. (2008a)	2008	Tonella and Ricca (2004a) and Bellettini et al. (2005)	Marchetto et al. (2008b), Tonella and Ricca (2004a) (one common author) and Bellettini et al. (2005) (one common author)
Cross-browser compatibility testing			
Choudhary et al. (2012)	2012	Mesbah and Prasad (2011) and Choudhary et al. (2010)	Choudhary et al. (2010) (two authors in common)
Support for testing			
Halfond et al. (2009)	2009	Halfond and Orso (2007), Deng et al. (2004), and a crawler called Spider	Halfond and Orso (2007) (two authors in common)

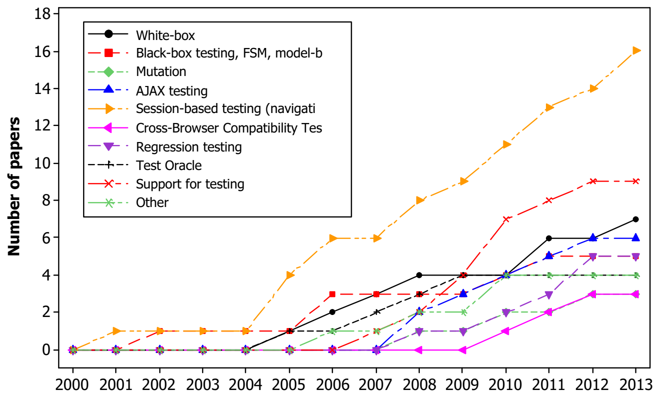


Fig. 13. Trend of the number of studies under each WAT theme area over the years.

that the time spent in Minamide and Engineering (2005)'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 Sprenkle et al. (2005a) was an empirical comparison of three test-suite reduction techniques for user-session-based testing: (1) concept analysis (Sampath et al., 2007), (2) the HGS requirements-based approach (Harrold et al., 1993), and (3) the Greedy requirements-based approach (Harrold et al., 1993). Note that the paper (Harrold et al., 1993) 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 Luo et al. (2009) 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 (Sampath et al., 2007) (based on concept analysis) and (Harrold et al., 1993) (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 Sampath et al. (2006a), 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 (Harrold et al., 1993) 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 Elbaum et al. (2005) 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 (2001), (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 Elbaum et al. (2005) 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 Sampath and Bryce (2008) 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. Sampath and Bryce (2008) borrowed the experimental objects and methodology from Sampath et al. (2007) and Sprenkle et al. (2005a, 2007). The goal of empirical study in Sampath and Bryce (2008) 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 Hao and Mendes (2006) describes two experiments that replicated Kallepalli and Tian's work (Kallepalli and Tian, 2001), 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. Our results showed that, in contrast to findings of Kallepalli and Tian (2001), multiple set UMMs were needed for trustworthy test case generation. In addition, the reliability assessment reported in Hao and Mendes (2006) corroborated results from Engström and Runeson (2011), confirming that UMMs seem to be a suitable testing mechanism to use to test web applications.

The study Dallmeier et al. (2013) 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 address of the application and, if necessary, user name and password. The empirical study reported in Dallmeier et al. (2013) 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 (Marchetto et al., 2008a,b; Mesbah and Van Deursen, 2009; Mesbah et al., 2012; Roest et al., 2010; Marchetto and Tonella, 2010), the only comparative study that we found in the theme area of AJAX testing is (Marchetto et al., 2008a). The authors of Marchetto et al. (2008a) had proposed a state-based testing technique for AJAX in an earlier work (Marchetto et al., 2008b). Marchetto et al. (2008a) reported a case study-based comparison of four types of testing

Table 24
Frequency of studies under each WAT theme area.

	Dynamic application		Static application	
	Validation	Evaluation	Validation	Evaluation
Server side	Marback et al. (2012), Sampath et al. (2005, 2006a, 2007), Halfond and Orso (2008), Sprenkle et al. (2005b, 2007, 2008, 2012), Harman and Alshahwan (2008), Dobolyi et al. (2010), Ran et al. (2009), Luo et al. (2009), Ozkinaci and Betin Can (2011), Ettema and Bunch (2010), Artzi et al. (2008), Thummalapenta et al. (2013), Halfond and Orso (2007), Artzi and Pistoia (2010), Halfond et al. (2009), Andrews et al. (2010), Liu and Kuan Tan (2008), Ricca and Tonella (2002b), Hao and Mendes (2006) and Peng and Lu (2011)	Sprenkle et al. (2005a, 2011), Alshahwan and Harman (2011), Elbaum et al. (2005), Sampath and Bryce (2008) and Sampath et al. (2006b)	Hao and Mendes (2006)	–
Client side				
Asynchronous (Ajax)	Artzi et al. (2011), Mesbah and Prasad (2011), Pattabiraman and Zorn (2010), Thummalapenta et al. (2013), Mirshokraie and Mesbah (2012), Amalfitano et al. (2010a), Marchetto et al. (2008b), Choudhary et al. (2010) and Dallmeier et al. (2013)	–	–	–
Synchronous	Saxena et al. (2010), Choudhary et al. (2012), Kallepalli and Tian (2001), Jensen and Møller (2011), Li et al. (2010), Sakamoto et al. (2013) and Tian and Ma (2006)	Marchetto et al. (2008a), Mirshokraie et al. (2013), Mesbah et al. (2012), Dobolyi and Weimer (2010), Roest et al. (2010), Marchetto and Tonella (2010)	Kallepalli and Tian (2001) and Tian and Ma (2006)	–
Both				
Asynchronous (Ajax)	–	–	–	–
Synchronous	Praphamontripong and Offutt (2010), Alshahwan et al. (2009, 2012) and Marchetto et al. (2007)	Mesbah and Van Deursen (2009)	–	–
Unspecified	Marchetto (2008)		–	–

techniques applied to AJAX web applications: (1) state-based testing (Marchetto et al., 2008b), (2) coverage-based/white-box testing (Tonella and Ricca, 2004a), (3) black-box (record and playback tools), and (4) UML model-based testing (Bellettini et al., 2005). 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 (Eldh et al., 2006). 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 (Mesbah and Prasad, 2011; Choudhary et al., 2010, 2012), there was one comparative study (Choudhary et al., 2012) which compared its proposed tool (called CrossCheck) to the two tools (CrossT and WebDiff) proposed in the two earlier studies (Mesbah and Prasad, 2011; Choudhary et al., 2010).

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 (Halfond et al., 2009) 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* (Halfond and Orso, 2007), *dfw* (Deng et al., 2004), 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). Fig. 14 shows the results for each attribute and Table 24 shows the combined picture.

As Fig. 14 shows, most of the empirical studies (62% of the studies, 36 papers) target the server side application, 26 studies (45%) target client side applications and only 5 studies (9%) provide

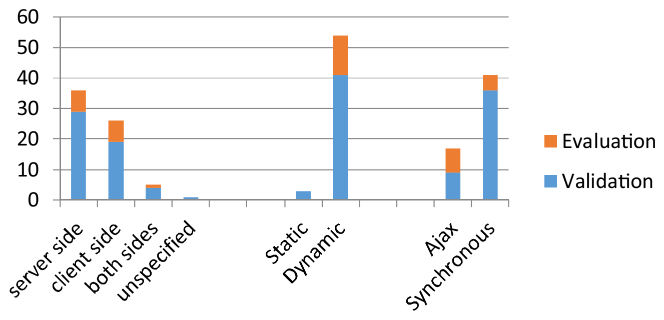


Fig. 14. Empirical studies break down on the attributes of web applications under test: location, dynamicity, and synchronicity.

Table 25
Empirical studies presenting results on client-tier web technologies.

	HTML	DOM	JavaScript
Validation	Artzi et al. (2011), Praphamontripong and Offutt (2010), Sprenkle et al. (2005b, 2007, 2008, 2012), Luo et al. (2009), Sampath et al. (2005), Alshahwan et al. (2012), Marchetto et al. (2007), Thummalapenta et al. (2013), Alshahwan et al. (2009), Mirshokraie and Mesbah (2012), Kallepalli and Tian (2001), Li et al. (2010), Sakamoto et al. (2013), Marchetto (2008), Hao and Mendes (2006), Tian and Ma (2006), Choudhary et al. (2010) and Dallmeier et al. (2013)	Mesbah and Prasad (2011), Choudhary et al. (2012), Pattabiraman and Zorn (2010), Thummalapenta et al. (2013), Mirshokraie and Mesbah (2012), Jensen and Møller (2011), Li et al. (2010), Sakamoto et al. (2013), Amalfitano et al. (2010a), Marchetto et al. (2008b), Marchetto (2008) and Choudhary et al. (2010)	Artzi et al. (2011), Saxena et al. (2010), Mesbah and Prasad (2011), Alshahwan et al. (2012), Choudhary et al. (2012), Pattabiraman and Zorn (2010), Marchetto et al. (2007), Alshahwan et al. (2009), Mirshokraie and Mesbah (2012), Jensen and Møller (2011), Marchetto et al. (2008b), Choudhary et al. (2010) and Dallmeier et al. (2013)
Evaluation	Marchetto et al. (2008a), Alshahwan and Harman (2011), Mesbah et al. (2012) and Dobolyi and Weimer (2010)	0	0

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, Fig. 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.

We also investigated how much evidence exists on the client-tier technology and on the server-tier technology. Table 25 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 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 (Marchetto and Tonella, 2010).

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

Table 26
Empirical studies presenting results on server-tier web technologies.

	PHP	J2EE	.Net	Perl/CGI	Other
Validation	Marback et al. (2012), Dobolyi et al. (2010), Alshahwan et al. (2012), Ettema and Bunch (2010), Marchetto et al. (2007), Artzi et al. (2008), Artzi and Pistoia (2010), Marchetto (2008) and Choudhary et al. (2010)	Sampath et al. (2007), Praphamontripong and Offutt (2010), Halfond and Orso (2008), Sprenkle et al. (2005b, 2007, 2012), Luo et al. (2009), Sampath et al. (2005, 2006a), Thummalapenta et al. (2013), Halfond et al. (2009), Amalfitano et al. (2010a), Marchetto et al. (2008b), Liu and Kuan Tan (2008) and Peng and Lu (2011)	Ozkinaci and Betin Can (2011) and Marchetto et al. (2007)	Ran et al. (2009)	–
Evaluation	Sprenkle et al. (2005a), Alshahwan and Harman (2011) and Mesbah et al. (2012)	Sprenkle et al. (2005a, 2011), Mesbah et al. (2012), Sampath and Bryce (2008), Marchetto and Tonella (2010) and Sampath et al. (2006b)	–	Elbaum et al. (2005)	Marchetto and Tonella (2010)

targeting some of these fault types have been proposed. However, 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.

- 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 35 and 28 studies, respectively. Construct and conclusion threats were identified in 9 and 6 studies only. Representativeness of injected faults/mutations was the most frequently identified type of internal validity threats (in 17 studies). Representativeness of SUTs was the most frequently identified type of external validity threats (in 30 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 been 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., real world case study.
- 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.

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 as complete a 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 (Garousi et al., 2013), 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 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 (Dogan et al., 2013). 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.

6. 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 paper we presented the first SLR of papers published between 2000 and 2013 in the area of web application functional testing (WAT), as a follow-up complementary study our recent SM study in the same domain. Our initial search retrieved 193 papers of which 95 were included in this study using a selection strategy. Based on a set of three research questions (RQs), data and empirical evidence were systematically extracted from the primary studies and were then synthesized to answer the RQs. 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: (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, and (7) evidence regarding the scalability of the WAT techniques. Our SLR shows the state-of-the-art in web application testing, areas that have been covered and techniques and 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.

In answering RQ 1.2, we extracted the types of fault models/bug taxonomies specific to web applications. We found 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.

To ensure readability and accessibility of the results obtained by the review, our online repository (hosted on the Google Docs system) (Dogan et al., 2013) supports backward cross-referencing by directly reporting how each analyzed paper is positioned with respect to each research question.

Acknowledgements

This work was supported by the Visiting Scientist Fellowship Program (#2221) of the Scientific and Technological Research Council of Turkey (TÜBİTAK). Vahid Garousi was additionally supported by the University of Calgary and the Natural Sciences and Engineering Research Council of Canada (NSERC).

Appendix A.

See Table 27.

Table 27
Details data for the list of fault models/bug taxonomies in the pool.

Study	Fault types
Marchetto et al. (2008a)	• Authentication • Multi-lingual • Functionality • Portability • Navigation • Asynchronous communication • Session • Form construction
Praphamontripong and Offutt (2010)	• Faults in simple link transitions • Faults in form link transitions • Faults in component expression transitions • Faults in operational transitions • Faults in redirect transitions
Sprenkle et al. (2007)	• 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
Sprenkle et al. (2005b)	• Link fault: changing a hyperlink's location
Luo et al. (2009)	• Multi-tier architecture faults • GUI faults • Hyperlinked structure faults • User authentication faults
Ozkinaci and Betin Can (2011)	• Syntactic HTML faults, e.g., element not allowed, missing attribute, end tag for unfinished element
Pattabiraman and Zorn (2010)	• 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
Mirshokraie et al. (2013)	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
Ettema and Bunch (2010)	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
Marchetto et al. (2007)	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

Table 27 (*Continued*)

Study	Fault types
Artzi et al. (2008)	• Faults specific to PHP: execution failures are caused by missing an included file, wrong MySQL query and uncaught exceptions • Producing malformed HTML
Mesbah et al. (2012)	• DOM validity • Back-button compatibility
Mirshokraie and Mesbah (2012)	• DOM modifications
Elbaum et al. (2005)	• 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
Kallepalli and Tian (2001)	• 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
Dobolyi and Weimer (2010)	• 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
Jensen and Møller (2011)	• 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
Artzi and Pistoia (2010)	• 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
Marchetto et al. (2008b)	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
Peng and Lu (2011)	• GUI faults • Database operation faults • Navigation faults
Choudhary et al. (2010)	• Layout issues • Differences in element position • Size • Visibility or appearance • Functionality issues

References[7]

Alalfi, M.H., Cordy, J.R., Dean, T.R., 2010. Automating coverage metrics for dynamic web applications. In: European Conference on Software Maintenance and Reengineering, pp. 51–60.

Alshahwan, N., Harman, M., 2011. Automated web application testing using search based software engineering. In: IEEEACM International Conference on Automated Software Engineering, pp. 3–12.

Alshahwan, N., Harman, M., Marchetto, A., Tonella, P., 2009. Improving web application testing using testability measures. In: IEEE International Symposium on Web Systems Evolution, pp. 49–58.

Alshahwan, N., Harman, M., Marchetto, A., Tiella, R., Tonella, P., Kessler, F.B., 2012. Crawlability metrics for web applications. In: IEEE International Conference on Software Testing, Verification and Validation, pp. 151–160.

Amalfitano, D., Fasolino, A.R., Tramontana, P., 2010, April. Rich internet application testing using execution trace data. In: International Conference on Software Testing, Verification, and Validation Workshops, pp. 274–283.

Amyot, D., Roy, J., Weiss, M., 2005. UCM-driven testing of web applications. In: Proceedings of the International SDL Forum, pp. 247–264.

Andrews, A.A., Offutt, J., Alexander, R.T., 2005. Testing web applications by modeling with FSMs. Software Systems Modeling 4 (3), 326–345.

Andrews, A.A., Offutt, J., Dyreson, C., Mallery, C.J., Jerath, K., Alexander, R., 2010. Scalability issues with using FSMWeb to test web applications. Information and Software Technology 52 (January (1)), 52–66.

Artzi, S., Pistoia, M., 2010, May. Practical fault localization for dynamic web applications. In: Proceedings of the 2nd ACM/IEEE International Conference on Software Engineering, vol. 1, pp. 265–274.

Artzi, S., Kiezun, A., Dolby, J., Tip, F., Dig, D., Paradkar, A., Ernst, M.D., 2008. Finding bugs in dynamic web applications. In: Proceedings of the International Symposium on Software Testing and Analysis, p. 261.

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

Bellettini, C., Marchetto, A., Trentini, A., Comelico, V., 2005. TestUml: user-metrics driven web applications testing. Proceedings of the ACM Symposium on Applied Computing, 1694–1698.

Benedikt, M., Freire, J., Godefroid, P., 2002. VeriWeb: automatically testing dynamic web sites. In: Proceedings of the International World Wide Web Conference (WWW).

Bordbar, B., Anastasakis, K., 2006. MDA and analysis of web applications. In: Trends in Enterprise Application Architecture. Springer, Berlin, Heidelberg, pp. 44–55.

Castelluccia, D., Mongiello, M., Ruta, M., Totaro, R., 2006. WAVer: a model checking-based tool to verify web application design. Electronic Notes in Theoretical Computer Science 157 (May (1)), 61–76.

Choudhary, S.R., Versee, H., Orso, A., 2010. WEBDIFF: automated identification of cross-browser issues in web applications. In: IEEE International Conference on Software Maintenance, pp. 1–10.

Choudhary, S.R., Prasad, M.R., Orso, A., 2012. Crosscheck: combining crawling and differencing to better detect cross-browser incompatibilities in web applications. In: International Conference on Software Testing, Verification and Validation, April, pp. 171–180.

Dallmeier, V., Burger, M., Orth, T., Zeller, A., 2013. WebMate: generating test cases for web 2.0. In: Software Quality. Increasing Value in Software and Systems Development, pp. 55–69.

Di Lucca, G.A., Fasolino, A.R., Tramontana, P., 2006. A technique for reducing user session data sets in web application testing. In: IEEE International Symposium on Web Site Evolution, pp. 7–13.

Di Sciascio, E., Donini, F.M., Mongiello, M., Totaro, R., Castelluccia, D., 2005. Design verification of web applications using symbolic model checking. In: Web Engineering. Springer, Berlin, Heidelberg, pp. 69–74.

[7] References of primary studies.

Dobolyi, K., Weimer, W., 2010. Modeling consumer-perceived web application fault severities for testing. In: Proceedings of the International Symposium on Software Testing and Analysis, p. 97.

Dobolyi, K., Soechting, E., Weimer, W., 2010. Automating regression testing using web-based application similarities. International Journal on Software Tools for Technology Transfer 13 (2), 111–129.

Elbaum, S., Rothermel, G., Karre, S., Ii, M.F., 2005. Leveraging user-session data to support web application testing. IEEE Transactions on Software Engineering 31 (3), 187–202.

Ernits, J., Roo, R., Jacky, J., Veanes, M., 2009. Model-based testing of web applications using NModel. In: Proceedings of the IFIP WG International Conference on Testing of Software and Communication Systems and International FATES Workshop, vol. 5826, pp. 211–216.

Ettema, T., Bunch, C., 2010. Eliminating navigation errors in web applications via model checking and runtime enforcement of navigation state machines. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering ACM, pp. 235–244.

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

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

Halfond, W.G.J., Orso, A., 2007. Improving test case generation for web applications using automated interface discovery. In: Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, p. 145.

Halfond, W.G.J., Orso, A., 2008. Automated identification of parameter mismatches in web applications. In: Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering, p. p181.

Halfond, W.G.J., Anand, S., Orso, A., 2009. Precise interface identification to improve testing and analysis of web applications. In: Proceedings of the International Symposium on Software Testing and Analysis, p. 285.

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

Harman, M., Alshahwan, N., 2008. Automated session data repair for web application regression testing. In: 2008 International Conference on Software Testing, Verification, and Validation, pp. 298–307.

Hu, W.I.C.C., 2002. Constructing an object-oriented architecture for web application testing. Journal of Information Science and Engineering 18 (1), 59–84.

Jensen, S.H., Møller, A., 2011. Modeling the HTML DOM and browser API in static analysis of JavaScript web applications. In: Proceedings of the ACM SIGSOFT Symposium and the European Conference on Foundations of Software Engineering, pp. 59–69.

Kallepalli, C., Tian, J., 2001. Measuring and modeling usage and reliability for statistical web testing. IEEE Transactions on Software Engineering 27 (11), 1023–1036.

Koopman, P., Achten, P., Plasmeijer, R., 2007. 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 4902 LNCS, 299–315.

Li, N., Xie, T., Jin, M., Liu, C., 2010. Perturbation-based user-input-validation testing of web applications. Journal of Systems and Software 83 (November (11)), 2263–2274.

Licata, D., Krishnamurthi, S., 2004. Verifying interactive web programs. In: Proceedings of the International Conference on Automated Software Engineering, pp. 164–173.

Liu, C.-H., 2006. Data flow analysis and testing of JSP-based web applications. Information and Software Technology 48 (December (12)), 1137–1147.

Liu, H., Kuan Tan, H.B., 2008. Testing input validation in web applications through automated model recovery. Journal of Systems and Software 81 (2), 222–233.

Liu, C.-H.L.C.-H., Kung, D.C., Hsia, P.H.P., Hsu, C.-T.H.C.-T., 2000. Structural testing of web applications. Proceedings of the International Symposium on Software Reliability Engineering, ISSRE 2000, 84–96.

Lucca, G.D., Fasolino, A., Faralli, F., 2002. Testing web applications. In: Proceedings of the International Conference on Software Maintenance, pp. 310–319.

Luo, X., Ping, F., Chen, M.-H., 2009. Clustering and tailoring user session data for testing web applications. In: International Conference on Software Testing Verification and Validation, pp. 336–345.

Mansour, N., Houri, M., 2006. Testing web applications. Information and Software Technology 48 (January (1)), 31–42.

Marback, A., Do, H., Ehresmann, N., 2012. An effective regression testing approach for PHP web applications. In: IEEE International Conference on Verification and Validation, pp. 221–230.

Marchetto, A., 2008. Talking about a mutation-based reverse engineering for web testing: a preliminary experiment. In: International Conference on Software Engineering Research, Management and Applications, pp. 161–168.

Marchetto, A., Tonella, P., 2010. Using search-based algorithms for Ajax event sequence generation during testing. Empirical Software Engineering 16 (December (1)), 103–140.

Marchetto, A., Ricca, F., Tonella, P., 2007. Empirical validation of a web fault taxonomy and its usage for fault seeding. In: IEEE International Workshop on Web Site Evolution, pp. 31–38.

Marchetto, A., Ricca, F., Tonella, P., 2008a. A case study-based comparison of web testing techniques applied to AJAX web applications. International Journal on Software Tools for Technology Transfer 10 (October (6)), 477–492.

Marchetto, A., Tonella, P., Ricca, F., 2008b. State-based testing of Ajax web applications. In: International Conference on Software Testing, Verification, and Validation, pp. 121–130.

Matos, E.C.B., Sousa, T.C., 2010. From formal requirements to automated web testing and prototyping. Innovations in Systems and Software Engineering 6 (January (1–2)), 163–169.

Mesbah, A., Prasad, M.R., 2011. Automated cross-browser compatibility testing. In: International Conference on Software Engineering, pp. 561–570.

Mesbah, A., Van Deursen, A., 2009. Invariant-based automatic testing of AJAX user interfaces. In: Proceedings of IEEE International Conference on Software Engineering, pp. 210–220.

Mesbah, A., Van Deursen, A., Roest, D., 2012. Invariant-Based Automatic Testing of Modern Web Applications. IEEE Transactions on Software Engineering 38 (1), 35–53.

Minamide, Y., Engineering, D.S., 2005. Static approximation of dynamically generated web pages. In: Proceedings of International Conference on World Wide Web, pp. 432–441.

Mirshokraie, S., Mesbah, A., 2012. JSART: JavaScript assertion-based regression testing. Web Engineering, vol. 1. Springer, Berlin, Heidelberg, pp. 238–252.

Mirshokraie, S., Mesbah, A., Pattabiraman, K., 2013. Efficient JavaScript mutation testing. In: IEEE International Conference on Software Testing, Verification and Validation.

Offutt, J., Wu, Y., 2009. Modeling presentation layers of web applications for testing. Software & Systems Modeling 9 (2), 257–280.

Offutt, J., Wu, Y., Du, X., Huang, H., 2004. Bypass testing of web applications. International Symposium on Software Reliability Engineering, 187–197.

Ozkinaci, M., Betin Can, A., 2011. Detecting execution and HTML errors in ASP.Net web applications. In: Proceedings of the 6th International Conference on Software and Data Technologies, pp. 172–178.

Pattabiraman, K., Zorn, B., 2010, November. DoDOM: leveraging DOM invariants for web 2.0 application robustness testing. In: IEEE International Symposium on Software Reliability Engineering, pp. 191–200.

Peng, X., Lu, L., 2011. User-session-based automatic test case generation using GA. Journal of Physical Sciences 6 (13), 3232–3245.

Praphamontripong, U., Offutt, J., 2010. Applying mutation testing to web applications. In: International Conference on Software Testing Verification and Validation Workshops, pp. 132–141.

Qi, Y., Kung, D., Wong, E., 2006. An agent-based data-flow testing approach for Web applications. Information and Software Technology 48 (12), 1159–1171.

Raffelt, H., Margaria, T., Steffen, B., Merten, M., 2008. Hybrid test of web applications with webtest. In: Proceedings of Workshop on Testing Analysis and Verification of Web Services and Applications, pp. 1–7.

Ran, L., Dyreson, C., Andrews, A., Bryce, R., Mallery, C., 2009. Building test cases and oracles to automate the testing of web database applications. Information and Software Technology 51 (2), 460–477.

Ricca, F., Tonella, P., 2001. Analysis and testing of Web applications. Proceedings of the 23rd International Conference on Software Engineering 47 (6), 25–34.

Ricca, F., Tonella, P., 2002a. Construction of the system dependence graph for web application slicing. In: Proceedings of the IEEE International Workshop on Source Code Analysis and Manipulation, pp. 123–132.

Ricca, F., Tonella, P., 2002b. Testing processes of web applications. Annals of Software Engineering 14 (1–4), 93–114.

Roest, D., Mesbah, A., Van Deursen, A., 2010. Regression testing Ajax applications: coping with dynamism. In: International Conference on Software Testing, Verification and Validation, pp. 127–136.

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

Sampath, S., Bryce, C., 2008. Prioritizing user-session-based test cases for web applications testing. In: International Conference on Software Testing, Verification, and Validation, vol. 1, pp. 141–150.

Sampath, S., Mihaylov, V., Souter, A., Pollock, L., 2004. Composing a framework to automate testing of operational web-based software. In: Proceedings of the 20th IEEE International Conference on Software Maintenance, pp. 104–113.

Sampath, S., Gibson, E., Sprenkle, S., Pollock, L., 2005. Coverage criteria for testing web applications. In: Computer and Information Sciences. University of Delaware (Tech. Rep., 2005-017).

Sampath, S., Sprenkle, S., Gibson, E., Pollock, L., 2006a. Integrating customized test requirements with traditional requirements in web application testing. In: Proceedings of the Workshop on Testing, Analysis, and Verification of Web Services and Applications, pp. 23–32.

Sampath, S., Sprenkle, S., Gibson, E., Pollock, L., 2006b. Web application testing with customized test requirements—an experimental comparison study. In: International Symposium on Software Reliability Engineering, pp. 266–278.

Sampath, S., Sprenkle, S., Gibson, E., Pollock, L., Greenwald, A.S., 2007. Applying concept analysis to user-session-based testing of web applications. IEEE Transactions on Software Engineering 33 (10), 643–658.

Saxena, P., Akhawe, D., Hanna, S., Mao, F., McCamant, S., Song, D., 2010. A symbolic execution framework for JavaScript. In: IEEE Symposium on Security and Privacy, pp. 513–528.

Song, B., Miao, H., 2009. Modeling web applications and generating tests: a combination and interactions-guided approach. In: IEEE International Symposium on Theoretical Aspects of Software Engineering, vol. 2007, pp. 174–181.

Sprenkle, S., Sampath, S.S.S., Gibson, E., Pollock, L., Souter, A., 2005a. An empirical comparison of test suite reduction techniques for user-session-based testing of

Web applications. In: IEEE International Conference on Software Maintenance, pp. 587–596.
Sprenkle, S., Gibson, E., Sampath, S., Pollock, L., 2005b. Automated replay and failure detection for web applications. In: Proceedings of the IEEEACM International Conference on Automated Software Engineering, pp. 253–262.
Sprenkle, S., Pollock, L., Esquivel, H., Hazelwood, B., Ecott, S., 2007. Automated Oracle comparators for testing web applications. In: IEEE International Symposium on Software Reliability, pp. 117–126.
Sprenkle, S., Esquivel, H., Hazelwood, B., Pollock, L., 2008. WEBVIZOR: A Visualization Tool for Applying Automated Oracles and Analyzing Test Results of Web Applications , pp. 89–93.
Sprenkle, S., Pollock, L., Simko, L., 2011. A study of usage-based navigation models and generated abstract test cases for web applications. In: IEEE International Conference on Software Testing Verification and Validation, pp. 230–239.
Sprenkle, S., Cobb, C., Pollock, L., 2012. Leveraging user-privilege classification to customize usage-based statistical models of web applications. In: IEEE International Conference on Software Testing, Verification and Validation, pp. 161–170.
Stepien, B., Peyton, L., Xiong, P., 2008. Framework testing of web applications using TTCN-3. International Journal on Software Tools for Technology Transfer 10 (April (4)), 371–381.
Thummalapenta, S., Lakshmi, K.V., Sinha, S., Sinha, N., Chandra, S., 2013. Guided test generation for web applications. Proceedings of the International Conference on Software Engineering, 162–171.
Tian, J., Ma, L.I., 2006. Web testing for reliability improvement. Advances in Computers 67, 178–225.
Tonella, P., Ricca, F., 2002. Dynamic model extraction and statistical analysis of Web applications. In: Proceedings International Workshop on Web Site Evolution, pp. 43–52.
Tonella, P., Ricca, F., 2004a. A 2-layer model for the white-box testing of Web applications. In: IEEE International Workshop on Web Site Evolution, pp. 11–19.
Tonella, P., Ricca, F., 2004b. Statistical testing of web applications. Journal of Software Maintenance and Evolution: Research and Practice 16 (12), 103–127.
Tonella, P., Ricca, F., 2005. Web application slicing in presence of dynamic code generation. Automated Software Engineering 12 (2), 259–288.
Törsel, A.M., 2013. A tool for web applications using a domain-specific modelling language and the NuSMV model checker. In: IEEE International Conference on Software Testing, Verification and Validation.
Xiong, W., Bajwa, H., Maurer, F., 2005. 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.
Zheng, Y., Bao, T., Zhang, X., Lafayette, W., 2011. Statically locating web application bugs caused by asynchronous calls. Distribution 195 (6), 805–814.

Further Reading

Afzal, W., Torkar, R., Feldt, R., 2008. A systematic mapping study on non-functional search-based software testing. In: International Conference on Software Engineering and Knowledge Engineering, pp. 488–493.
Afzal, W., Torkar, R., Feldt, R., 2009. A systematic review of search-based testing for non-functional system properties. Information and Software Technology 51, 957–976.
Alalfi, M.H., Cordy, J.R., Dean, T.R., 2009. Modelling methods for web application verification and testing: state of the art. Software Testing, Verification & Reliability 19, 265–296.
Ali, S., Briand, L.C., Hemmati, H., Panesar-Walawege, R.K., 2010. A systematic review of the application and empirical investigation of searchbased test case generation. IEEE Transactions on Software Engineering 36, 742–762.
Amalfitano, D., Fasolino, A., Tramontana, P., 2010b. Techniques and tools for rich internet applications testing. In: Proceedings IEEE International Symposium on Web Systems Evolution, pp. 63–72.
Andrews, J.H., Briand, L.C., Labiche, Y., Namin, A.S., 2006. Using mutation analysis for assessing and comparing testing coverage criteria. IEEE Transactions on Software Engineering 32, 608–624.
Banerjee, I., Nguyen, B., Garousi, V., Memon, A., 2013. Graphical user interface (GUI) testing: systematic mapping and repository. Information and Software Technology 55, 1679–1694.
Barmi, Z.A., Ebrahimi, A.H., Feldt, R., 2011. Alignment of requirements specification and testing: a systematic mapping study. In: Proceedings of the IEEE Fourth International Conference on Software Testing, Verification and ValidationWorkshops, pp. 476–485.
Business Internet Group San Francisco (BIG-SF), 2003. Black Friday Report on Web Application Integrity. The Business Internet Group of San Francisco http://www.tealeaf.com/news/press_releases/2003/0203.asp.
Binder, R.V., 1996. Testing object-oriented software: a survey. In: Proceedings of the Tools-23: Technology of Object-Oriented Languages and Systems, p. p374.
Bozkurt, M., Harman, M., Hassoun, Y., 2010. Testing Web Services: A Survey, Technical Report TR-10-01. Department of Computer Science, King's College, London.
Canfora, G., Penta, M.D., 2008. Service-oriented architectures testing: a survey. In: International Summer Schools on Software Engineering, pp. 78–105.
Cooper, H., Hedges, L.V., Valentine, J.C., 2009. The Handbook of Research Synthesis and Meta-Analysis, 2nd ed. Russell Sage Foundation, New York.
Cruzes, D.S., Dybå, T., 2010. Synthesizing evidence in software engineering research. In: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement.
Cruzes, D.S., Dybå, T., 2011a. Recommended steps for thematic synthesis in software engineering. In: Proc. International Symposium on Empirical Software Engineering and Measurement, pp. 275–284.
Cruzes, D.S., Dybå, T., 2011b. Research synthesis in software engineering: a tertiary study. Information and Software Technology 53, 440–455.
Deng, Y., Frankl, P., Wang, J., 2004. Testing web database applications. ACM SIGSOFT Software Engineering Notes 29, 1–10.
Dogan, S., Betin-Can, A., Garousi, V., 2013. Online SLR Data for Web Application Testing (WAT), http://goo.gl/VayAr (Accessed 15.07.13).
Elberzhager, F., Münch, J., Nha, V.T.N., 2012. A systematic mapping study on the combination of static and dynamic quality assurance techniques. Information and Software Technology 54, 1–15.
Eldh, S., Hansson, H., Punnekkat, S., Pettersson, A., Sundmark, D., 2006. A framework for comparing efficiency, effectiveness and applicability of software testing techniques. In: Testing: Academic & Industrial Conference-Practice and Research Techniques, pp. 159–170.
Endo, A.T., Simao, A.d.S., 2010. 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, p. 89.
Engström, E., Runeson, P., 2011. Software product line testing—a systematic mapping study. Journal of Information and Software Technology 53, 2–13.
Engström, E., Runeson, P., Skoglund, M., 2010. A systematic review on regression test selection techniques. Journal of Information and SoftwareTechnology 53, 14–30.
Feldt, R., Magazinius, A., 2010. Validity threats in empirical software engineering research—an initial survey. In: Proceedings of the Software Engineering and Knowledge Engineering Conference.
Garousi, V., 2012a. Online Paper Repository for Software Test-Code Engineering: A Systematic Mapping, http://www.softqual.ucalgary.ca/projects/SM/STCE (Accessed 5.12.12).
Garousi, V., 2012b. Online Paper Repository for GUI Testing, http://www.softqual.ucalgary.ca/projects/2012/GUI_SM/ (Accessed 5.12.12).
Garousi, V., 2012c. Online Paper Repository for Developing Scientific Software: A Systematic Mapping, http://softqual.ucalgary.ca/projects/2012/SM_CSE/ (Accessed 1.12.12).
Garousi, V., Mesbah, A., Betin-Can, A., Mirshokraie, S., 2013. A systematic mapping of web application testing. Elsevier Journal of Information and Software Technology 55, 1374–1396.
Grindal, M., Offutt, J., Andler, S.F., 2005. Combination testing strategies: a survey. Software Testing, Verification, and Reliability 15.
Harrold, M.J., Gupta, R., Soffa, M.L., 1993. A methodology for controlling the size of a test suite. ACM Transactions on Software Engineering and Methodology 2, 270–285.
Hellmann, T.D., Hosseini-Khayat, A., Maurer, F., 2010. Agile interaction design and test-driven development of user interfaces—a literature review. In: Agile Software Development: Current Research and Future Directions.
Insfran, E., Fernandez, A., 2008. A systematic review of usability evaluation in web development. In: Proceedings of Web Information Systems Engineering, pp. 81–91.
Ivarsson, M., Gorschek, T., 2011. A method for evaluating rigor and industrial relevance of technology evaluations. Empirical Software Engineering 16, 365–395.
Jia, Y., Harman, M., 2011. An analysis and survey of the development of mutation testing. IEEE Transactions of Software Engineering 37, 649–678.
Jia, Y., Harman, M., 2013. Mutation Testing Paper Repository, http://www.dcs.kcl.ac.uk/pg/jiayue/repository (Accessed July 2003).
Juristo, N., Moreno, A.M., Vegas, S., 2004. Reviewing 25 years of testing technique experiments. Empirical Software Engineering 9, 7–44.
Kam, B., Dean, T.R., 2009. Lessons learned from a survey of web applications testing. In: Proceedings of the International Conference on Information Technology: New Generations.
Kitchenham, B., Charters, S., 2007. Guidelines for performing systematic literature reviews in software engineering. In: Evidence-Based Software Engineering.
Liborg, N.K., Masters thesis 2004. A Study of Threats to Validity in Controlled Software Engineering Experiments. University of Oslo https://www.duo.uio.no/handle/10852/9155.
Lucca, G.A.D., Fasolino, A.R., 2006. Testing web-based applications: the state of the art and future trends. Information and Software Technology 48, 1172–1186.
McMinn, P., 2004. Search-based software test data generation: a survey: research articles. Software Testing, Verification and Reliability 14, 105–156.
Memon, A.M., Nguyen, B.N., 2010. Advances in automated model-based system testing of software applications with a GUI front-end. Advances in Computers 80, 121–162.
Neto, P.A.D.M.S., Machado, I.D.C., McGregord, J.D., Almeida, E.S.D., Meira, S.R.D.L., 2011a. A systematic mapping study of software product lines testing. Information and Software Technology 53, 407–423.
Neto, P.A.d.M.S., Runeson, P., Machado, I.d.C., Almeida, E.S.d., Meira, S.R.d.L., Engström, E., 2011b. Testing software product lines. IEEE Software 28, 16–20.
Neto, C.R.L., Neto, P.A.D.M.S., Almeida, E.S.D., Meira, S.R.D.L., 2012. A mapping study on software product lines testing tools. In: Proceedings of International Conference on Software Engineering and Knowledge Engineering.
Palacios, M., García-Fanjul, J., Tuya, J., 2011. Testing in service oriented architectures with dynamic binding: a mapping study. Information and Software Technology 53, 171–189.
Păsăreanu, C.S., Visser, W., 2009. A survey of new trends in symbolic execution for software testing and analysis. International Journal on Software Tools for Technology Transfer 11, 339–353.

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

Shull, F., Singer, J., Sjøberg, D.I.K., 2008. Guide to Advanced Empirical Software Engineering. Springer, London.

Souza, S.R.S., Brito, M.A.S., Silva, R.A., Souza, P.S.L., Zaluska, E., 2011. Research in concurrent software testing: a systematic review. In: Proceedings of the Workshop on Parallel and Distributed Systems: Testing, Analysis, and Debugging, pp. 1–5.

Waliaa, G.S., Carverb, J.C., 2009. A systematic literature review to identify and classify software requirement errors. Information and Software Technology 51, 1087–1109.

Weyuker, E.J., 1993. Can we measure software testing effectiveness? In: Proc. of IEEE Software Metrics Symposium, pp. 100–107.

Zakaria, Z., Atan, R., Ghani, A.A.A., Sani, N.F.M., 2009. Unit testing approaches for BPEL: a systematic review. In: Proceedings of the Asia-Pacific Software Engineering Conference, pp. 316–322.

Zhang, Y., 2013. Repository of Publications on Search based Software Engineering, http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/ (Accessed July 2013).

Serdar Doğan is a software developer in Aselsan A.S. Ankara, Turkey. During his career, he has participated in several software engineering projects at Scientific & Technological Research Council of Turkey (TUBITAK) and Aselsan. Serdar received his BSc from Hacettepe University's Computer Engineering Department in 2004, Ankara, Turkey and completed his M.Sc. degree at Middle East Technical University's Information Systems Department, Ankara, Turkey in 2013.

Aysu Betin-Can is an associate professor in Informatics Institute at Middle East Technical University, Ankara Turkey. Her research interests include web application testing, automated software verification, concurrent software. She received the ACM SIGSOFT Distinguished Paper Award in 2005 and the Best Paper Award at the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005).

Vahid Garousi is an associate professor at the University of Calgary, Canada and a visiting professor in the Middle East Technical University (METU) in Ankara, Turkey. He is also a practicing software engineering consultant and coach, and has provided consultancy and corporate training services in several countries in the areas of software testing and quality assurance, model-driven development, and software maintenance. During his career, Vahid has been active in initiating a number of software testing and software engineering projects in Canada. Vahid completed his PhD in Software Engineering in Carleton University, Canada, in 2006. He has been involved as an organizing or program committee member in many software engineering conference, such as ICST, ICSP, CSEE&T, and MoDELS. He is a member of the IEEE and the IEEE Computer Society, and is also a licensed professional engineer (PEng) in the Canadian province of Alberta. He has been selected a Distinguished Visitor (speaker) for the IEEE Computer Society's Distinguished Visitors Program (DVP) for the period of 2012–2015. Among his awards is the Alberta Ingenuity New Faculty Award in June 2007. For more information, visit: www.ucalgary.ca/~vgarousi.