Contents lists available at SciVerse ScienceDirect

Information and Software Technology

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

A systematic mapping study of web application testing

Vahid Garousi [a,c,*], Ali Mesbah [b], Aysu Betin-Can [c], Shabnam Mirshokraie [b]

[a] Electrical and Computer Engineering, University of Calgary, Calgary, Canada
[b] Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada
[c] Informatics Institute, Middle East Technical University, Ankara, Turkey

ARTICLE INFO

Article history:
Received 14 June 2012
Received in revised form 9 February 2013
Accepted 11 February 2013
Available online 1 March 2013

Keywords:
Systematic mapping
Web application
Testing
Paper repository
Bibliometrics

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 147 papers in the area of web application testing, which have appeared between 2000 and 2011.
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 in this specialized field.
Method: We review and structure the body of knowledge related to web application testing through a systematic mapping (SM) study. As part of this study, we pose two sets of research questions, define selection and exclusion criteria, and systematically develop and refine a classification schema. In addition, we conduct a bibliometrics analysis of the papers included in our study.
Results: Our study includes a set of 79 papers (from the 147 retrieved papers) published in the area of web application testing between 2000 and 2011. We present the results of our systematic mapping study. Our mapping data is available through a publicly-accessible repository. We derive the observed trends, for instance, in terms of types of papers, sources of information to derive test cases, and types of evaluations used in papers. We also report the demographics and bibliometrics trends in this domain, including top-cited papers, active countries and researchers, and top venues in this research area.
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 systematic mapping can help researchers to obtain an overview of existing web application testing approaches and indentify areas in the field that require more attention from the research community.

Contents

* Corresponding author at: Electrical and Computer Engineering, University of Calgary, Calgary, Canada. Tel.: +1 403 210 5412.
E-mail addresses: vgarousi@ucalgary.ca, vahid@metu.edu.tr (V. Garousi), amesbah@ece.ubc.ca (A. Mesbah), betincan@metu.edu.tr (A. Betin-Can), shabnamm@ece.ubc.ca (S. Mirshokraie).

0950-5849/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2013.02.006

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, and (3) universal access from any machine connected to the Internet. On the downside, the use of server and browser technologies make web applications particularly error-prone and challenging to test, causing serious dependability threats. In addition to financial costs, errors in web applications result in loss of revenue and credibility.

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

According to Petersen et al. [47], a systematic mapping (SM) is a method to review, classify, and structure papers related to a specific research field in software engineering. The goal is to obtain an overview of existing approaches, outlining the coverage of the research field in different facets of the classification scheme. Identified gaps in the field serve as a valuable basis for future research directions [39,36]. Results of SM studies can also be valuable resources for new researchers (e.g., PhD students) by providing a detailed overview of a specific research area [16].

There are major differences between SM studies and systematic literature reviews (SLR). Kitchenham et al. [39] report a comprehensive comparison of SM and SLR studies using the following seven criteria: goals, research questions, search process, scope, search strategy requirements, quality evaluation, and results. According to that report, 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. Research questions 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, research questions of a SLR are specific, meaning that they are related to outcomes of empirical studies. For example, they could be of the form: Is technology/method A better than B? Unlike a SLR [37], finding evidence for impact of a proposed approach is not the main focus in a systematic mapping [47]. An SLR analyzes primary studies, reviews them in depth and describes their methodology and results. SLRs are typically of greater depth than SMs. Often, SLRs include an SM as a part of their study [47]. In other words, the results of a SM can be fed into a more rigorous SLR study to support evidence-based software engineering [38].

SM studies generally consist of five steps [47] including a definition of research questions, conducting the search for relevant papers, screening of papers, keywording of abstracts, and data extraction and mapping, which we follow in this paper.

To the best of our knowledge, this paper is the first systematic mapping study in the area of web application testing. The main contributions of this paper are:

- A generic classification scheme for categorizing papers in the field of web application testing.
- A systematic mapping study in the field of functional testing of web applications, structuring related research work over the past decade by capturing and analyzing 79 included papers.
- An analysis of the demographic trends and bibliometrics in the area of web application testing.
- An online repository of the papers collected and analyzed through this systematic study.

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 provides the classification scheme we have developed for the web testing domain and the process used for constructing it. Section 5 presents the results of the systematic mapping followed by the bibliometric analysis in Section 6. Section 7 discusses the main findings, implications and trends. Finally, Section 8 concludes the paper.

2. Related work

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

2.1. Secondary studies in software testing

We were able to find 22 secondary studies reported, as of this writing, in different areas of software testing. We list these studies in Table 1 along with some of their attributes. For instance, the "number of papers" (No.) included in each study shows the number of primary studies analyzed, which varies from six (in [32]) to 264 (in [33]). Our study analyzes 147 papers and includes 79 in the final pool, as described in Section 3.

We have observed that SMs and SLRs have recently started to appear in the area of software testing. We found six SMs in the area of software testing: (1) product lines testing [20,23,44], (2) SOA testing [45], (3) requirements specification and testing [12], and (4) non-functional search-based software testing [3]. There are also five SLRs in the area: (1) search-based non-functional testing [4], (2) search-based test-case generation [7], (3) formal testing of web services [22], (4) unit testing approaches for Business Process Execution Language (BPEL) [51], and (5) regression test selection techniques [24]. The remaining 11 studies are "surveys", "taxonomies", "literature reviews", and "analysis and survey", terms used by the authors themselves to describe their secondary studies [13,35,42,31,17,46,14,33,19,43,32]. Note that none of these studies is related to web application testing, which is the focus of our study.

2.2. Online paper repositories in SE

A few recent secondary studies have reported online repositories to supplement their study with the actual data. These repositories are the by-products of SM studies and will be useful to practitioners by providing a summary of all the works in a given area. Most of these repositories are maintained and updated regularly, typically every six months. For instance, Harman et al. have developed and shared two online paper repositories: one in the area of mutation testing [33,34], and another in the area of search-based software engineering (SBSE)[1] [52].

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 our mapping study as an online paper repository, which we intend to update on a regular basis.

2.3. Secondary studies in web application testing

Here we provide a brief overview of existing secondary studies (e.g., surveys/taxonomy papers), focusing on different areas of web testing and analysis.

Di Lucca and Fasolino [21] present 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. Alalfi et al. [5] present a survey of 24 different modeling methods used in web verification and testing. The authors categorize, compare and analyze the different modeling methods according to navigation, behavior, and content. Amalfitano et al. [8] propose 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. Van Deursen and Mesbah [49] describe the challenges of testing Ajax-based web applications, discuss to what extent current automated testing can be used to address those challenges, and formulate directions for future research. Marin et al. [40] discuss the testing challenges of future web applications and provide a concise overview of current testing techniques and their limitations for testing future web applications.

All these existing studies have several shortcomings that limit their replication, generalization, and usability in structuring the research body on web application testing. 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.

To the best of our knowledge, there are currently no systematic mappings or reviews for the field of web application testing.

3. Research methodology

This systematic mapping is carried out following the guidelines and process proposed by Petersen et al. [47], which has the following main steps:

- Defining research questions.
- Defining the search query and searching for papers.

[1] This repository is quite comprehensive and has 1,020 papers as of April 2012.

Table 1
18 Secondary studies in software testing.

Type	Secondary study area	No.	Year	Ref.
SM	Non-func. search-based testing	35	2008	[3]
	SOA testing	33	2011	[45]
	Requirements specification	35	2011	[12]
	Product lines testing	45	2011	[20]
	Product lines testing	64	2011	[23]
	Product lines testing tools	N/A[a]	2015	[44]
SLR	Search-based non-func. testing	35	2009	[4]
	Unit testing for BPEL	27	2009	[51]
	Formal testing of web services	37	2010	[22]
	Search-based test-case generation	68	2010	[7]
	Regression test selection techniques	27	2010	[24]
Survey/Analysis	Object oriented testing	140	1996	[13]
	Testing techniques experiments	36	2004	[35]
	Search-based test data generation	73	2004	[41]
	Combinatorial testing	30	2005	[31]
	SOA testing	64	2008	[17]
	Symbolic execution	70	2009	[46]
	Testing web services	86	2010	[14]
	Mutation testing	264	2011	[33]
	Product lines testing	16	2011	[19]
Taxonomy	Model-based GUI testing	33	2010	[43]
Lit. rev.	TDD of user interfaces	6	2010	[32]

[a] We could not find/access this paper.

- Screening the retrieved papers, resulting in a set of relevant papers.
- Keywording using abstracts, resulting in a classification scheme.
- Data extraction and mapping process, resulting in a systematic map.

The process that we have used to conduct this SM study is outlined in Fig. 1. The process starts with article selection (discussed in detail in Section 3.2). Then, the classification scheme/map is systematically built to classify the primary studies (Section 4). Afterwards, the systematic mapping itself is conducted and results are reported in Section 5. Trends and demographics of studies are then analyzed and reported in Section 6.

In the remainder of this section, we explain our (1) goal and research questions and (2) paper selection strategy.

3.1. Goal and research questions

The goal of our study is to identify, analyze, and synthesize work published during the past ten years in the field of *web application testing*. We aim to (1) systematically review related scientific papers in the field in order to conduct a mapping of the area and (2) present bibliometrics and demographic analysis of the field.

Based on our research goal, we formulate two main research questions (RQ 1 and RQ 2). To extract detailed information, each question is further divided into a number of sub-questions, as described below.

- **RQ 1 – systematic mapping:** What is the research space of the literature in the field of functional testing and analysis of web applications? The sub-questions of this RQ are:
  - **RQ 1.1 – type of contribution:** How many papers present test methods/techniques, test tools, test models, test metrics, or test processes? The SM guideline paper by Petersen et al. [47] proposes the above types of contributions. Answering this RQ will enable us to assess whether the community as a whole has had more focus on developing new test techniques, or, more focus on developing new test tools.
  - **RQ 1.2 – type of research method:** What type of research methods are used in the papers in this area? The SM guideline paper by Petersen et al. [47] proposes the following types of research methods: solution proposal, validation research, and evaluation research. The rationale behind this RQ is that knowing the breakdown of the research area with respect to (w.r.t.) research-facet types will provide us with the maturity of the field in using empirical approaches.
  - **RQ 1.3 – type of testing activity:** What type(s) of testing activities are presented in the papers? Examples of testing activity are: test-case design, test automation, and test execution [9]. Addressing this RQ will help us gain knowledge about the type of test activities that have been more popular.
  - **RQ 1.4 – test location:** How many client-side versus server-side testing approaches have been presented? Is it true that server-side testing has received more focus? Addressing this RQ will help us determine whether client-side or server-side testing methods have been more popular.
  - **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? The term test "artifacts" in this context denotes any type of artifact generated and used for purpose of web-application testing, e.g., test cases, test requirements, test harness, test code, etc. Example sources of information to derive test artifacts include (but are not limited to): source code (white-box testing), requirements such as models (black-box testing), invariants and web usage logs. The rationale behind this RQ is that knowing the breakdown of various testing techniques w.r.t. their inputs will enable practising web developers and researchers to use the most appropriate techniques given the availability of inputs in their projects.
  - **RQ 1.7 – technique to derive test artifacts:** What techniques have been used to generate test artifacts? Examples are: requirements based testing, static code analysis, and coverage. Addressing this RQ will help practising web developers and researchers, when searching for test techniques, to select the appropriate techniques based on their web testing needs.

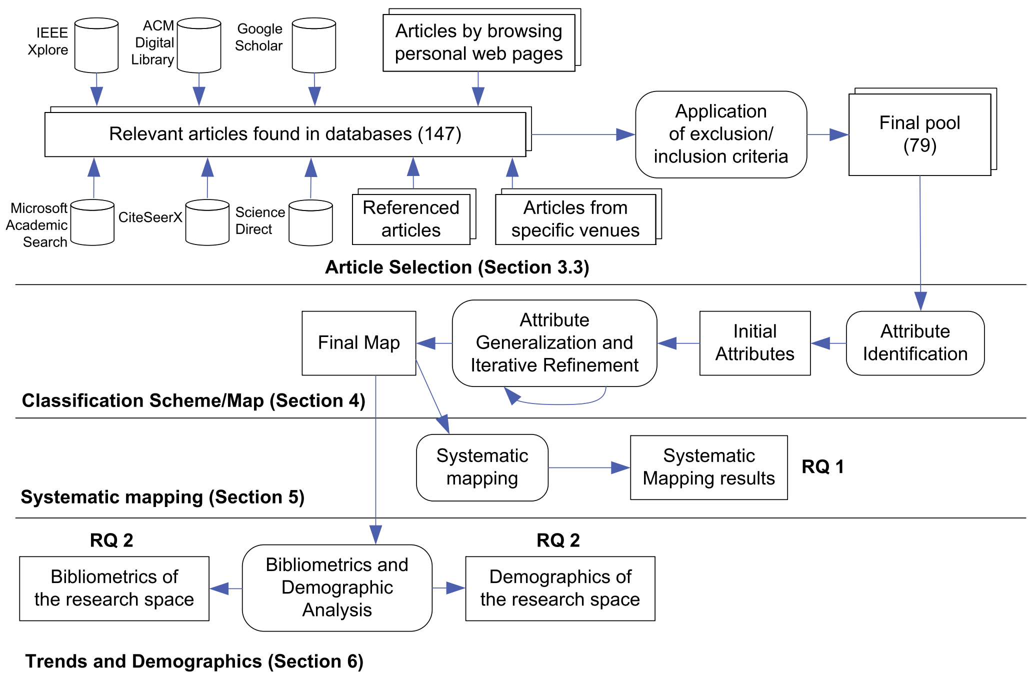

IEEE Xplore
ACM Digital Library
Google Scholar
Articles by browsing personal web pages
Relevant articles found in databases (147)
Application of exclusion/ inclusion criteria
Final pool (79)
Microsoft Academic Search
CiteSeerX
Science Direct
Referenced articles
Articles from specific venues
Article Selection (Section 3.3)
Final Map
Attribute Generalization and Iterative Refinement
Initial Attributes
Attribute Identification
Classification Scheme/Map (Section 4)
Systematic mapping
Systematic Mapping results
RQ 1
Systematic mapping (Section 5)
RQ 2
Bibliometrics of the research space
Bibliometrics and Demographic Analysis
RQ 2
Demographics of the research space
Trends and Demographics (Section 6)

Fig. 1. The research protocol used in this study.

- **RQ 1.8 – type of test artifact:** Which types of test artifacts (e.g., test cases, test inputs) have been generated? Similar to the above RQ's, addressing this RQ will again help practising web developers and researchers to select the proposed techniques easily, based on their web testing needs.
- **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? For example, some papers use mutation analysis while some use coverage measurement to assess the applicability and effectiveness of their proposed web testing techniques.
- **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**: The following set of RQs have been motivated by reviewing the existing bibliometrics studies in software engineering, e.g., [48,27,30,29].
  - **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 2.4 – active countries:** Which countries are contributing the most to this area, based on the affiliations of the researchers?
  - **RQ 2.5 – top venues:** Which venues (i.e., conferences, journals) are the main targets of papers in this field?

Sub-questions RQ 1.1–1.16 together will help us answer the first main question (RQ 1). Similarly, in order to answer RQ 2 properly, we need to address sub-questions RQ 2.1–2.5.

3.2. Paper selection strategy

Our paper selection strategy consists of the following activities:

1. Resource selection, search query definition, and searching.
2. Application of exclusion and inclusion criteria.

We explain each step subsequently below. We then, present an overview of the final pool of papers and the online repository that were produced after conducting the above activities.

3.2.1. Resource selection and search query definition

To find relevant papers, we searched the following six major online academic paper search engines: (1) IEEE Xplore,[2] (2) ACM Digital Library,[3] (3) Google Scholar,[4] (4) Microsoft Academic Search,[5] (5) CiteSeerX,[6] and (6) Science Direct.[7] These search engines have also been used in other similar studies [25,20,3].

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

In order to ensure that we were including as many relevant publications as possible in the pool of papers, all authors identified and proposed potential search keywords in several iterations. The coverage landscape of this SM is the area of *functional* testing of web applications, as well as (dynamic or static) analysis to support web-application testing. 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 follows:

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

Related papers published between 2000 and 2011 were included in our pool. 2000 is the year that the very first web testing papers appeared. Note that the paper selection phase of this study was carried out during the Summer 2011 (May until August) and, thus, papers published by the end of that summer were included in our pool.

To decrease the risk of missing related and important publications, similar to previous systematic mapping/review studies, the authors looked for:

- Related papers by browsing personal web pages of active researchers in the area.
- Related papers referenced from papers already in the pool.
- Related papers from major software (e.g., TSE, ICSE, FSE, ICST, ISSTA) and web (e.g., ICWE, WSE, WWW, TWEB, TOIT) engineering research venues.

3.2.2. Exclusion and inclusion criteria

Since the focus on this study is on functional testing, a large number of papers that target non-functional properties, such as accessibility and performance testing or security vulnerability detection (e.g., cross-site scripting), were excluded from our study. We included papers with static analysis used an enabling technique in web application testing.

To increase the reliability of our study and its results, the authors applied a systematic voting process among the team members in the paper selection phase for deciding whether to include or exclude any of the papers in the first version of the pool. This process was also utilized to minimize personal bias of each of the authors. The team members had conflicting opinions on four papers, which were resolved through discussions.

Our voting mechanism (i.e., exclusion and inclusion criteria) was based on two questions: (1) Is the paper relevant to functional web application testing and analysis? (2) Does the paper include a relatively sound validation? These criteria were applied to all papers, including those presenting techniques, tools, or case studies/experiments. Each author then independently answered each of the two questions for each paper. Only when a given paper received at least two positive answers (from three voting authors) for each of the two questions, it was included in the pool. Otherwise, it was excluded.

We primarily voted for papers based on their title, abstract, keywords, as well as their evaluation sections. If not enough information could be inferred from the abstract, a careful review of the contents was also conducted to ensure that all the papers had a direct relevance to our focused topic. We considered all ***peer-reviewed*** papers regardless of the venue. As such, we considered papers published in journals, conference proceedings, workshops, and magazines.

Only papers written in English and only those available electronically were included. If a conference paper had a more recent journal version, only the latter was included. We excluded papers on “web services”, because the nature of web services differs from that of web applications.

3.3. Final pool of papers and the online repository

Initially, our pool included 147 papers. After the exclusion criteria were applied, the paper pool size decreased to 79 papers. The entire pool of 79 papers has been published as an online repository on the Google Docs service [26]. The intention is to update the online repository at least annually to add related papers that appear in the future. Detailed classification of each paper is also available in our online repository.

4. Classification scheme

To conduct a systematic mapping, a classification scheme (also called systematic map or attribute framework [18]) needs to be derived by a careful analysis of the primary studies [47]. Our classification scheme started with an initial version, and evolved during data extraction, through attribute generalization and iterative refinement steps. New categories were added, and existing categories were merged or split. The iterative refinement process was finalized and a stable final scheme was derived when the scheme was able to consistently categorize all the papers in the pool. This phase of our process is also depicted in our research protocol (See Fig. 1).

For this step, we collaboratively used an online spreadsheet in Google Docs to document the data extraction process. Each identified category of the classification scheme was added to the spreadsheet. When we entered the data of a paper into the scheme, we provided a short rationale why the paper should be in a certain category (for example, why/how a paper has applied evaluation research). We used the “observer triangulation” method in designing the classification scheme and data extraction (mapping) phases. Each paper was reviewed by at least two reviewers (authors of this paper) and differences of opinions were discussed in detail until a final decision was made. When needed, the classification scheme was also updated.

Table 2 shows our final classification scheme along with the research questions (RQs) addressed by each attribute of the map. Attributes of our classification scheme are discussed next.

The columns of the table show the research question (RQs), attributes, and possible types of the attribute. Also, the last two columns indicate whether for each of the attributes, multiple or just one type(s) can apply, respectively. For example, for RQ 1.1 (the contribution facet attribute), multiple types can be selected for the same paper. On the other hand, for the row corresponding to RQ 1.11 (static web sites versus dynamic web applications), only one type (static or dynamic) can be chosen, which is self explanatory.

We have adopted the two “type” attributes widely used in other SM studies: *contribution facet*, and *research facet*.

A contribution facet (corresponding to RQ 1.1) denotes the type of contribution(s) proposed in each paper and can be either: method/technique, tool, model, metric, process, or other [47]. Naturally, these contribution facets would turn to the following in our web application testing context: test method/technique, test tool, test model, test metric, and test process, respectively.

Table 2
The classification scheme developed and used in our study.

RQ	Attribute	Possible types	Multiple selections	Single selection
RQ 1				
RQ 1.1	Type of Paper-Contribution Facet	{Test method/technique, Test tool, Test model, Metric, Process, Other}	×	
RQ 1.2	Type of Paper-Research Facet	{Solution Proposal, Validation Research, Evaluation Research, Experience Papers, Philosophical Papers, Opinion Papers, Other}	×	
RQ 1.3	Type of Testing Activity	{Test-case Design (Criteria-based), Test-case Design (Human knowledge-based), Test Automation, Test Execution, Test Evaluation (oracle), Other}	×	
RQ 1.4	Testing Location	{Client-side, Server-side}	×	
RQ 1.5	Testing Level	{Unit, Integration, System}	×	
RQ 1.6	Source of information to derive Test artifacts	{Source code (white-box), Requirements and Source code (gray-box), Requirements (Models, etc.), Invariants, (user) logs, Inferred Model (automatic), Inferred Model (manual), Other}	×	
RQ 1.7	Techniques used	{Requirements based, Symbolic execution, Static code analysis, Dynamic code analysis, Coverage, Crawling, Concolic testing, Model checking, Search-based testing, Record/playback, Model-based, Other}	×	
RQ 1.8	Type of test artifact generated	{Test cases, Test input (data), Test requirements (not input values), Expected outputs (oracle), Test driver (code), Other}	×	
RQ 1.9	Manual versus Automated testing	{Manual, Automated}	×	
RQ 1.10	Type of the Evaluation Method	{Coverage (code, model), Mutation testing (fault injection), Manual comparison, Time/performance, Detecting real faults, Example, Other}	×	
RQ 1.11	Static Web Sites versus Dynamic Web Apps	{Static, Dynamic}		×
RQ 1.12	Synchronicity of HTTP calls	{Synchronous calls, Asynchronous calls (AJAX)}		×
RQ 1.13	Client-tier Web Technology	{HTML, DOM, JavaScript, N/A, Unknown, Other}	×	
RQ 1.14	Web-server-tier Web Technology	{PHP, J2EE, .NET, Ruby, Python, Perl (CGI), Java (Servlet, JSP), N/A, Unknown, Other}	×	
RQ 1.15	Presented tool(s)	Tool name: String Whether the tool is available for download (as reported in the paper): Boolean		
RQ 1.16	Attributes of the web software SUT(s)	# of SUTs: Integer SUT names: String[] Total SUT Size in LOC: Integer Other size metrics (e.g., # of pages, # of forms, etc.): Integer[] For each SUT, SUT scale ∈ {Academic experimental, Open-source, Commercial}		
RQ 2				
RQ 2.1	Publication year	Year: Integer		
RQ 2.2	Number of times the paper is cited in other papers	Number of Citations: Integer		
RQ 2.3	List of authors	Authors: String[]		
RQ 2.4	Venue in which the paper has appeared	Venue: String		
RQ 2.5	The country (ies) of the author(s)' affiliation	Author Countries: String[]		

The research facet attribute corresponds to RQ 1.2. As discussed by Petersen et al., research facet denotes the type of research approach used in each paper. We adopted the following research facets for our study:

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 is 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 lab.
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).
4. Experience Papers: Experience papers explain how something has been done in practice. It has to be the personal experience of the author(s).

Papers with examples only are categorized in (1), papers having validation sections, but not in the full-scale of systematic empirical studies, are categorized in (2), if the proposed technique in a study is evaluated comprehensively using systematic empirical evaluations (e.g., case study, controlled experiment), and its benefits, drawbacks, and threats to validity of the results are discussed thoroughly, we categorize its research facet as (3). Papers that merely report applications or experiences in practice are categorized in (4).

The next attribute in Table 2 is the type of *testing activity* proposed in each paper (corresponding to RQ 1.3). In their book on software testing [9], Ammann and Offutt divide testing activities into six types as follows: (1) test-case design based on criteria (e.g., line coverage), (2) test-case design based on human knowledge (e.g., exploratory testing), (3) test automation: embedding test values into executable test code (scripts), (4) test execution: running tests on the software under test and recording the results, (5) test evaluation (test oracle): evaluating results of testing (pass/fail), a.k.a. test verdict, and reporting results to developers, and (6) other. We found this particular classification applicable to the papers in our pool and adopted it for our classification scheme.

The *testing location* attribute (corresponding to RQ 1.4) can be client-side and/or server-side. Some papers present testing techniques for client side aspects of web applications, and others focus merely on the server side. There are also papers that test both client and server sides.

The next attribute denotes the *testing level*, which corresponds to RQ 1.5. As it has been discussed and defined in software testing books, e.g. [9], testing level in this context denotes, in an abstract viewpoint, the scope (granularity) of testing which could be: modules in isolation (unit testing), testing the integration of different modules (integration testing), or the entire software system (system testing). Conceptually, many alternative testing terminologies would also fall into each of the above categories, e.g., acceptance testing and even smoke testing fall into the system testing level.

RQ 1.6 is about the *source of information* to derive test artifacts (e.g., test cases): source code, requirements and source code (gray-box testing), requirements only (models, etc.), invariants, (user) logs, inferred models (derived automatically), inferred models (derived manually), and other.

Similar to RQ 1.6, the attributes and type sets for RQs 1.7 through 1.14 in Table 2 were derived iteratively. However, we should clarify the notion of 'test requirements' for RQ 1.8 (type of test artifact generated). Test requirements are usually not actual test input values, but the conditions that can be used to generate test inputs [9]. For example, a coverage-based test technique might require that a certain control-flow path of a function be covered, e.g., the set of logical expressions that should be made true. By having test requirements, one can manually or automatically generate test inputs, an area of research referred to as test-data generation [41].

In RQ 1.10, we looked for *evaluation methods* used in each paper. The evaluation methods we looked for included code or model coverage measurements, mutation testing, detecting real faults, and manual comparison. In addition, there were papers that provided examples and proofs of concept. One of the methods to evaluate a testing technique was measuring the time/performance of the testing effort. Note that, this time/performance attribute was an evaluation metric of a testing technique itself, and *not* the performance evaluation of an SUT by a proposed technique.

To answer RQ 1.15, we extracted the names of the web testing *tools* presented in each paper, and also checked whether the tool is available for download (as indicated in the paper).

For RQ 1.16, we extracted the following data regarding the web software under test (SUT) in each paper (i.e., used for validation/evaluation, a.k.a. subject systems): (1) number of SUTs, (2) SUT name(s), (3) total SUT size in Lines of Code (LOC), and (4) any other size metrics when reported (e.g., number of web pages, number of web forms). The scale of each SUT was determined according to the following categorization: (a) academic experimental, (b) open-source, or (c) commercial web application.

The attributes and type sets for RQs 2.1 through 2.5 relate to demographic and bibliometric information of papers and the explanations provided in Table 2 should be self-explanatory.

Building the classification scheme was an iterative process. We started with an initial version and then used attribute generalization and iterative refinement, while reviewing the papers in the pool, to derive the final classification scheme. The classification scheme was considered "final" if it was able to classify all the papers properly.

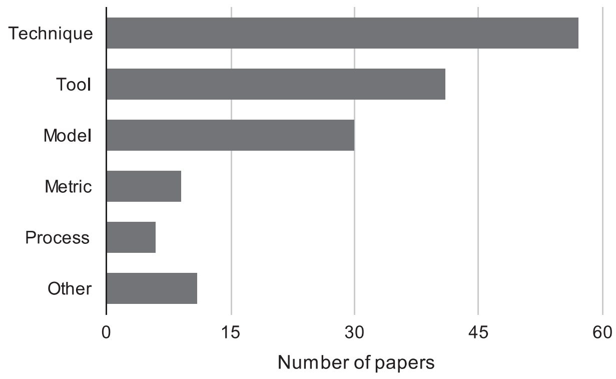
After we developed the classification scheme, the papers in the pool were then classified using the scheme, i.e., the actual data extraction took place. From the final table in the spreadsheet, we were then able to calculate the frequencies of publications in each category, presented in detail in Section 5.

5. Systematic mapping results

In this section, we present the results of our systematic mapping study (RQ 1).

5.1. RQ 1.1 – Types of papers by contribution facet

Fig. 2 shows the distribution of the type of papers by contribution face, for all the 79 papers included in our study. Based on their contributions, some papers were classified under more than one facet. For example, [60] made three contributions: (1) a test method (a framework for feed-back directed testing of JavaScript applications), (2) a test tool called *Artemis*, and (3) a test model (event-driven execution model). Fig. 3 depicts a histogram of the frequency of contribution facets for a single paper. Most papers presented two contribution facets, followed by only one contribution, three, and four contribution facets. There were five papers [96,115,74,94,84] that covered four facets. For example, [96] contributed: (1) a test method (automated cross-browser



Technique: [60, 111, 116, 104, 103, 96, 78, 54, 68, 105, 100, 88, 115, 130, 85, 117, 101, 102, 128, 72, 62, 122, 67, 75, 77, 97, 70, 66, 81, 82, 71, 99, 61, 110, 109, 56, 59, 95, 98, 131, 126, 83, 90, 74, 108, 89, 86, 58, 64, 57, 94, 84, 65, 127, 124]
Tool: [60, 116, 104, 103, 96, 78, 119, 54, 53, 68, 105, 115, 130, 101, 102, 128, 72, 62, 67, 75, 77, 55, 97, 80, 61, 109, 56, 59, 98, 126, 74, 108, 86, 64, 57, 94, 84, 65, 118, 129]
Test model: [125, 60, 87, 106, 96, 105, 115, 123, 85, 117, 72, 97, 66, 81, 82, 71, 99, 80, 109, 95, 90, 74, 89, 86, 64, 57, 94, 84, 124]
Metric: [111, 87, 53, 115, 93, 77, 55, 99, 74, 94]
Process: [96, 68, 115, 67, 108, 58]
Other: [92, 120, 121, 119, 91, 112, 69, 76, 73, 79, 113]

Fig. 2. Contribution facet. (See above mentioned references for further information).

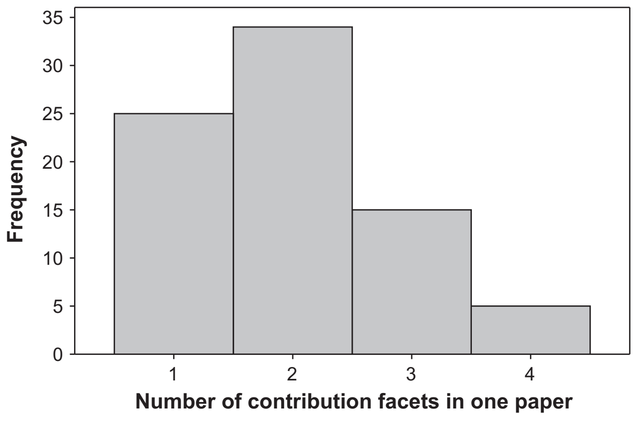


Fig. 3. Histogram of frequency of contribution facets per paper.

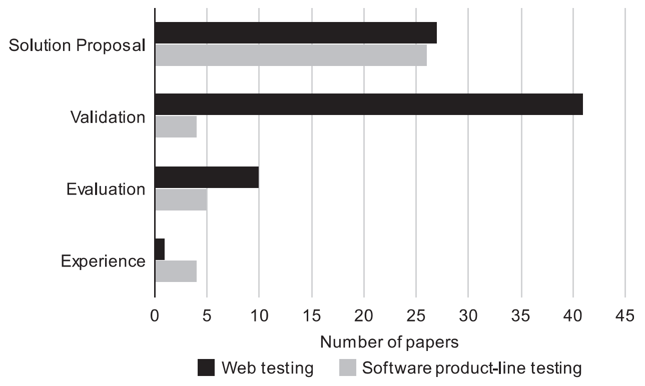
compatibility testing), (2) a test tool called *CrossT*, (3) a test model (state navigation model), and (4) a model generation process to facilitate web testing.

Fig. 2 indicates that proposing new techniques or improving an existing technique has attracted the most research with 54 papers (about 68%) focusing on this aspect. Also, relatively a high proportion (about 51%) of papers (36 out of 79) proposed web testing tools of some sort. Section 5.15 provides an extended discussion on test tools presented in the papers. There were 11 papers which could not be categorized into the five contribution facets of our scheme, thus we categorized them under 'Other'. Those papers were mainly secondary studies, such as comparison studies (e.g., [92]) or empirical studies (e.g., [120,121]).

The annual trend of the same data is shown in Fig. 4. It can be seen that in recent years, there is a focus on a mix of different facets. In terms of time, the earliest papers in our paper pool were published in 2001. This was the time that the web started to be widely used. We notice that except for a few exceptions (in 2003, 2006, 2007 and 2009), there is a clear increase in the quantity of papers on web application testing over the past decade. The simple reason is that web application and their testing are becoming more important as time goes by. Recall from Section 3.2.1 that since the paper selection of this study was done during the Summer 2011, only papers published (or accepted for publication) by the end of Summer 2011 are included in our pool. Thus, the data for 2011 is incomplete.

5.2. RQ 1.2 – Types of papers by research facet

Fig. 5 shows the types of papers by research facet. To provide a cross-subject comparison of these facets, we have compared our data with the facet distribution of a related recent systematic mapping which has been done in the area of software product-lines testing [20]. The research in web application testing is dominated by solution proposals (27 papers, 34.18%) and validation studies (41 papers, 51.90%). Since there are various types and locations of faults in web applications, there exists a large body of studies



Solution Proposal: [125, 116, 87, 104, 100, 115, 130, 107, 123, 85, 117, 122, 67, 75, 66, 82, 71, 99, 98, 73, 89, 86, 58, 64, 84, 65, 127, 129]
Validation Research: [60, 111, 106, 114, 103, 96, 78, 119, 54, 53, 68, 105, 88, 93, 101, 102, 128, 72, 91, 62, 77, 55, 112, 81, 80, 61, 76, 56, 59, 95, 131, 126, 83, 90, 74, 108, 57, 79, 124, 118]
Evaluation Research: [92, 120, 121, 97, 70, 69, 110, 109, 94, 113]
Experience Papers: [63]

Fig. 5. Research facet. (See above mentioned references for further information).

proposing different testing strategies and techniques. There were also a reasonable share of papers with full-scale experimental studies (10 papers, 12.66%). Comparing our data to the mapping results of the software product-lines testing literature [20], it is clear that, in terms of ratio, validation and evaluation studies are more popular in the web testing community compared to software product-lines testing. This indicates the relatively higher level of attention to empirical approaches in web application testing research.

The yearly trend of research facet types is shown in Fig. 6. The figure shows that in earlier years (from 2001 to 2006), more solution proposals with less rigorous empirical studies were published in the area. However from 2006 to 2011, we notice more rigorous empirical studies compared to solution-proposal-type papers. This is good news for the web testing community as it indicates that the empirical maturity of the literature in this area is increasing as a whole. Certainly, to understand the landscape of empirical

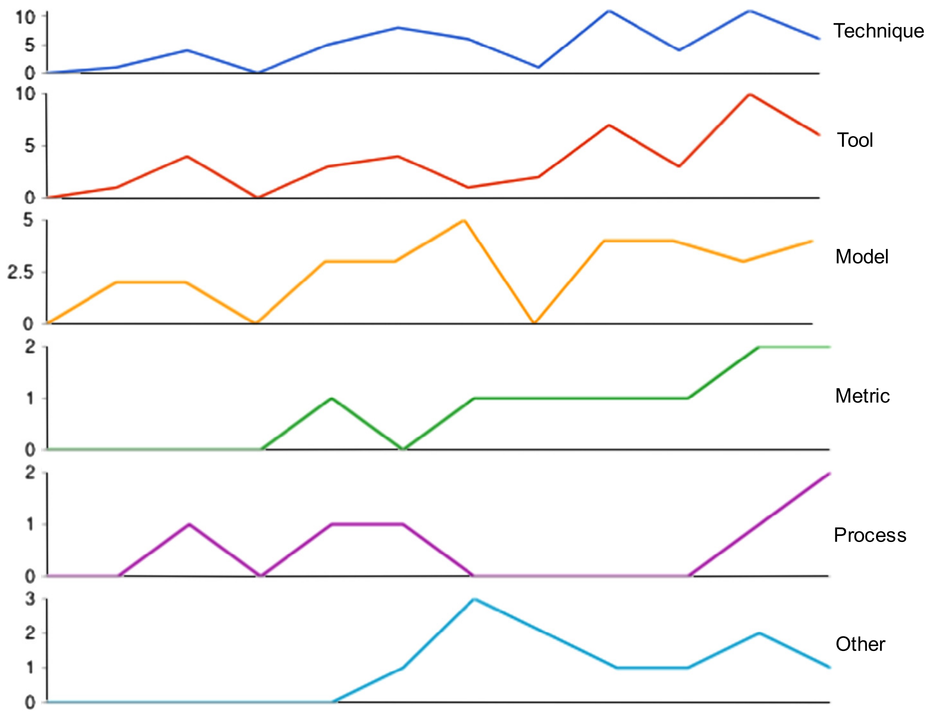


Fig. 4. Annual trend of papers by contribution facet. The *x*-axis denotes the years (2000–2011) and the *y*-axis is the number of papers.

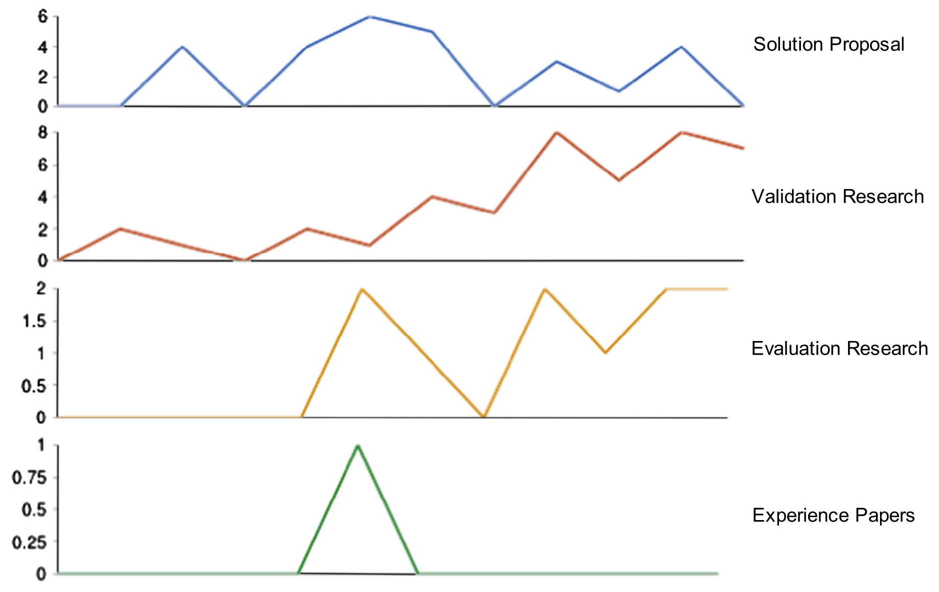


Fig. 6. Annual trend of papers by research facet. The *x*-axis denotes the years (2000–2011) and the *y*-axis is the number of papers.

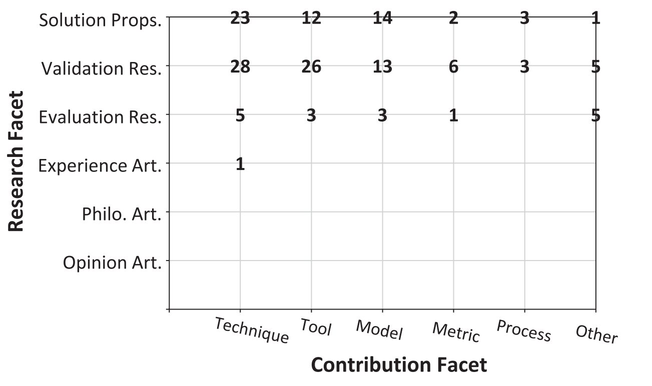
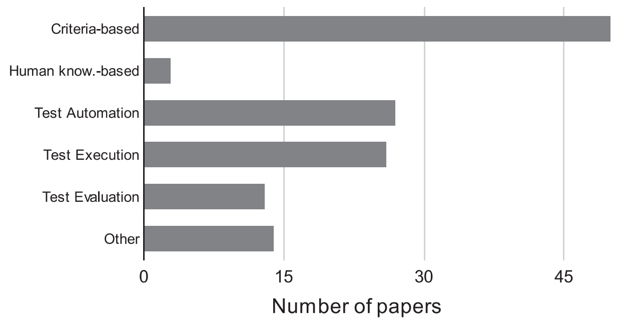


Fig. 7. Cross analysis of number of studies in different contribution facet types versus research facet types.



Test-case Design (Criteria-based): [125, 92, 60, 111, 120, 87, 104, 106, 114, 103, 96, 53, 105, 100, 88, 115, 130, 123, 93, 85, 101, 128, 91, 77, 97, 70, 81, 82, 99, 61, 109, 56, 59, 95, 98, 126, 83, 90, 73, 74, 108, 89, 86, 58, 64, 57, 79, 94, 113, 124]
Test-case Design (Human knowledge-based): [67, 75, 90]
Test Automation: [92, 60, 87, 104, 103, 96, 115, 130, 93, 101, 122, 67, 75, 97, 71, 99, 109, 59, 95, 126, 89, 86, 64, 57, 84, 65, 129]
Test Execution: [92, 60, 111, 106, 114, 96, 68, 115, 130, 93, 102, 62, 122, 75, 97, 71, 61, 109, 95, 126, 108, 86, 64, 84, 65, 129]
Test Evaluation (oracle): [119, 68, 105, 115, 130, 62, 97, 66, 82, 71, 109, 86, 118]
Other: [87, 78, 54, 107, 102, 72, 91, 55, 112, 80, 76, 84, 113, 124]

Fig. 8. Type of testing activity.

research in web testing, dedicated systematic reviews focusing on empirical studies in this area, for instance similar to a recent review [6] conducted in the search-based testing community, are needed.

Furthermore, we conducted a cross analysis of number of studies in different contribution facet types versus research facet types. Fig. 7 shows the results, in which the *x*-axis and *y*-axis show the different contribution facet and research facet types, respectively. For example, there are 23 studies which propose (contribute) a (test) "technique" and their research facet type is "solution proposal". As we can observe, the concentration in the top left corner of the chart is the highest, denoting that majority of the works are contributing techniques or tool and are low or medium in terms of research facet maturity.

5.3. RQ 1.3 – Type of testing activity

As mentioned in Section 4, we base our test activity categorization on the terminology proposed by Ammann and Offutt [9].

In the pool of 79 papers included in our mapping, 69 papers (87.3%) utilized code or model coverage. Fifty of those papers (63.3%) actually presented (coverage) criteria-based testing activities as seen in Fig. 8. Control-flow (e.g, line, branch coverage), data flow (e.g., all uses coverage), and model/state-based coverage (e.g, all nodes, or all paths criteria) customized for web applications have been proposed and evaluated in many papers, e.g, [125,92].

Only three papers (3.8%) presented test-case design based on human knowledge. This type of test-case design activity is usually advocated by the exploratory testing community [50] and also context-driven school of testing [2]. Thus, it seems that there is a relatively small number of researchers using exploratory testing for web applications. This is somewhat surprising since exploratory testing is quite popular in the software industry.

Test automation was another popular research activity as 27 papers (34.2%) address test automation. For example, 14 papers (e.g., [67,96,71]) adopted the popular browser automation tool *Selenium* [15] in their tools and evaluations. Ref. [94] approached test automation by using another existing tool called *FitNesse*. In [97], an approach for generating JUnit test cases from an automatically reverse-engineered state model was proposed.

A large portion of the papers, i.e., 26 of 79 (32.9%), addresses the test execution activity, which is merely concerned with running test cases on the software and recording the results. "Test harness" is also used by some, e.g., [71] as an acronym for test execution.

The test evaluation activity is also known as the test oracle concept. 13 papers (16.5%) addressed this issue. For example, [118] presented a tool for visualization of automated test oracles and test results. Ref. [109] proposed an approach for detecting and visualizing regressions in Ajax-based web applications. Ref. [66] generated test oracles in the form of Object Constraint Language (OCL) rules. Ref. [97] proposed DOM-based invariants as test oracles. Ref. [91] presented an automated test oracle, which determines the occurrence of fatal failures or HTML well-formedness failures (the latter via use of an HTML validator). Ref. [105] generated oracles in the context of database-driven web applications using Prolog rules.

There are also a number of papers that fall in other types of test activities than the five major types we have adopted from [9]. For instance, [87,121,112,56,127,113] were among the works that address test suite minimization (reduction) in web applications. We also found other topics such as: repairing test data [54], testability [55], reliability and statistical testing [81], fault localization [61], and test-case prioritization [110].

5.4. RQ 1.4 – Test locations

As far as the location is concerned, the majority of the papers have focused on testing the server-side. As Fig. 9 depicts, 41 (51.9%) of the papers focus on server-side of web applications, 22 (27.8%) targets the client side, and 14 (17.7%) examine both sides.

The numbers show that both ends of web applications are getting attention from researchers. Server-side of web applications usually has a database and consists of scripts or programs that generate HTML code to be interpreted by the client browsers. This complexity led many researchers to examine the server-side of the applications. On the other hand, with the rise of the client-side scripting (e.g., using JavaScript) and Ajax for building Web 2.0 applications, the complexity of web clients has increased. This in turn has motivated many researchers to shift the focus on client-side analysis and testing in recent years (as shown in Fig. 10).

Details and mapping of client-tier and server-tier web technologies will be discussed in Sections 5.13 and 5.14, respectively.

We also wanted to know whether more researchers are focusing towards or away from server-side versus client-side testing. Fig. 10 is a cumulative distribution function (CDF) and depicts the annual trend for the two types of test location. We can observe from this trend that, in terms of cumulative numbers, server-side testing papers out number the client-side testing papers. However, after year 2008 in terms of growth rate, the number of papers on client-side testing have been slightly higher than those on server-side testing. Quantitatively, the cumulative number of papers on client-side testing increased from nine in 2008 to 22 in 2011 (a growth of 144%), while the cumulative number of papers on server-side testing increased from 30 in 2008 to 41 in 2011 (a growth of only 37%).

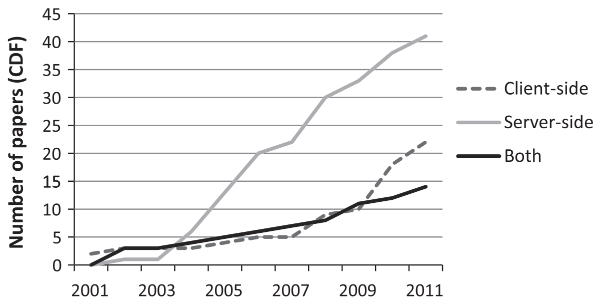
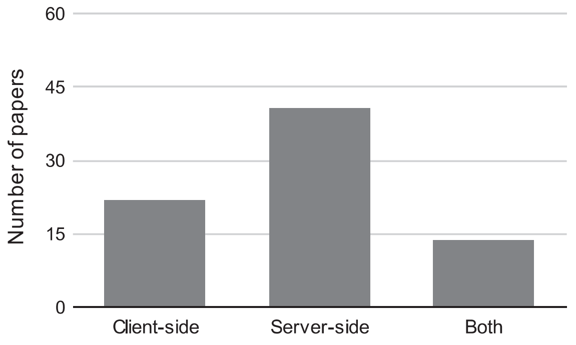


Fig. 10. Annual trend for test location (cumulative distribution function).



Client-side: [92, 60, 116, 106, 96, 123, 117, 102, 67, 75, 81, 82, 69, 80, 109, 56, 95, 83, 73, 94, 65, 124]
Server-side: [111, 120, 87, 121, 114, 78, 119, 54, 53, 68, 105, 100, 88, 115, 85, 101, 128, 72, 62, 122, 77, 112, 70, 99, 61, 76, 110, 59, 98, 126, 74, 108, 89, 58, 57, 79, 84, 127, 113, 118, 129]
Both: [125, 104, 103, 130, 107, 93, 91, 55, 97, 71, 131, 90, 86, 64]

Fig. 9. Test locations.

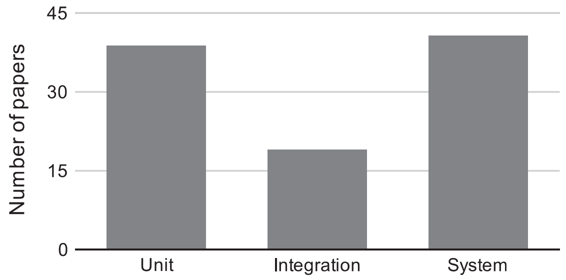
5.5. RQ 1.5 – Testing levels

To examine which test levels have received more attention, we have divided the papers into three groups: unit testing, integration testing, and system testing which is usually performed through a graphical user interface (GUI). The distribution is shown in Fig. 11.

In this context, a "unit" included a single HTML file, a source-code function inside a JavaScript (JS), JSP or PHP file. A paper was considered to have had a focus on web integration testing, if it addressed the problem of how to test several "units" together, e.g., the client–server interactions of a HTML or JS file on the client with the corresponding PHP handler module on server. Last but not least, a given paper was considered to have had a focus on web system testing, if it addressed testing of a given web software from a holistic standpoint, e.g., GUI testing of a web site.

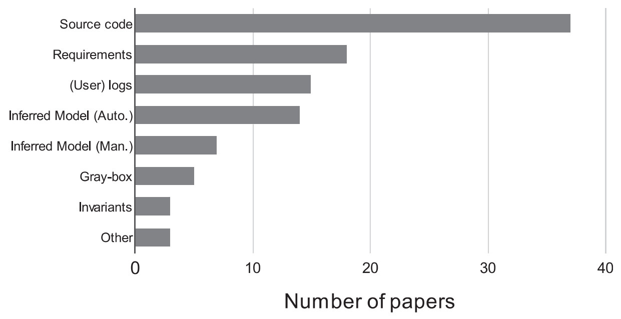
Unit testing and system testing have been almost equally popular (38% each) while integration testing has been targeted only in 24% of the papers.

Our main criteria to classify papers in these three categories were based on the coverage criteria and granularity of test approaches. For example, [111] performed client-side web-page-level, statement coverage measurement. It also analysed test coverage of functions and function calls. Thus, it was mapped to unit and integration testing levels. As another example, [78] tests each HTML document separately, thus we consider that as a unit testing approach.



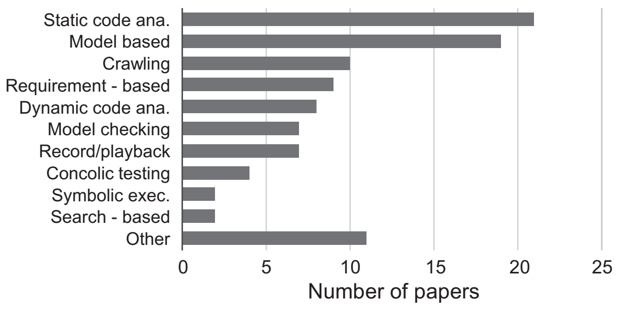
Unit testing: [92, 111, 104, 106, 114, 103, 96, 78, 119, 53, 123, 93, 85, 101, 128, 91, 63, 62, 67, 77, 55, 112, 97, 70, 66, 71, 80, 61, 76, 109, 56, 95, 131, 83, 74, 86, 64, 94, 113]
Integration testing: [111, 104, 106, 114, 103, 54, 105, 88, 85, 117, 72, 91, 55, 71, 99, 108, 89, 86, 127]
System testing: [125, 92, 60, 120, 87, 121, 96, 68, 100, 88, 115, 130, 93, 117, 102, 91, 122, 75, 55, 97, 70, 66, 81, 82, 71, 99, 110, 59, 98, 126, 90, 73, 58, 64, 57, 79, 84, 65, 124, 118, 129]

Fig. 11. Testing levels.



Source code (white-box): [125, 92, 60, 116, 104, 106, 103, 78, 54, 53, 68, 100, 115, 130, 107, 93, 85, 101, 128, 62, 122, 77, 55, 70, 99, 80, 61, 76, 98, 131, 74, 108, 89, 86, 84, 127, 118]
Requirements and Source code (gray-box): [117, 72, 112, 64, 84]
Requirements (Models, etc.) : [92, 121, 106, 78, 105, 117, 67, 66, 81, 71, 80, 73, 86, 57, 79, 113, 124]
Invariants : [102, 97, 66]
(User)logs : [111, 120, 87, 121, 114, 54, 88, 115, 70, 81, 110, 126, 79, 113, 124]
Inferred Model (automatic) : [92, 96, 75, 97, 109, 56, 95, 126, 90, 64, 94, 84, 65]
Inferred Model (manual) : [81, 82, 56, 59, 90, 108, 58]
Other: [119, 61, 95, 94, 83]

Fig. 12. Source of information to derive test artifacts.



Symbolic execution : [116, 76]
Static code analysis: [106, 78, 53, 68, 100, 107, 85, 122, 77, 55, 99, 80, 95, 98, 131, 74, 89, 64, 84, 118]
Dynamic code analysis, e.g., Coverage : [125, 60, 104, 53, 130, 70, 89, 127]
Crawling: [96, 54, 93, 55, 97, 109, 56, 126, 83, 65]
Concolic testing : [101, 128, 62, 61]
Model checking: [106, 117, 72, 66, 98, 84]
Search-based testing : [92, 94]
Record/playback: [111, 114, 115, 130, 102, 75, 112]
Model-based: [92, 120, 121, 54, 88, 81, 82, 71, 80, 59, 95, 126, 74, 108, 86, 58, 64, 57, 124]
Other: [111, 87, 104, 121, 106, 114, 103, 115, 123, 89, 79]

Fig. 13. Techniques used/proposed.

5.6. RQ 1.6 – Source of information to derive test artifacts

Fig. 12 shows the distribution of the type of information sources for all the 79 papers surveyed.

Thirty-seven papers (46.8%) used source code as information source to derive test artifacts. We discuss four example works [53,100,122,77] next. The coverage of PHP and SQL is analyzed to derive test cases in [53]. Ref. [100] analyzed client code to test server code. Ref. [122] worked on HTML source code. Ref. [77] used server-side source code to derive a set of interfaces for the purpose of interface testing.

The second mostly-used source to derive test artifacts is requirements (17 papers, or 21.5%), both textually and graphically represented. For example, [105] used state transition diagrams. Ref. [81] used state transitions in unified Markov models. Ref. [57] used use-case maps (UCMs). Last but not least, [66] derived test cases from a formal notation called Abstract Description of Interaction (ADI), a type of class diagram.

Four papers used both requirements and source code, thus following a gray-box approach. For example, in [84], properties to be verified (test oracle) were from requirements, but the test model to generate test inputs were generated from source code.

The next information source was user logs (15 out of 79, or 18%). For example, [124,81] generated unified Markov models from logs and then used those models to generate test cases. Ref. [79] analyzed server access and error logs to derive navigation patterns.

User-log-based test techniques are followed by inferred models, which range from state machines to UML models.

We further divided inferred models as automatically-inferred (16.5%) and manually-inferred (8.9%) models. Examples of approaches based on the automatically-inferred test models are the followings. Ref. [65] inferred a navigational model through crawling. Refs. [64,95] reverse engineered UML models from code. Ref. [92] inferred models from execution states. As an example of approaches based on the manually-inferred test models, Ref. [90] used user reports manually, collected during the testing phase, to refine a specific test model.

Finally, invariants are used in three papers (3.7%) in our pool. Ref. [66] expected invariants to be provided in form of Object Constraint Language (OCL) rules. Ref. [97] automatically crawled and dynamically asserted invariants on web applications. Ref. [102] derived DOM invariants dynamically by recording a sequence of user interactions with the application and observing the changes to the DOM tree by repeatedly replaying the sequence.

The remaining five papers (6.33%) used other source of information to derive test artifacts. For instance, the technique reported in this work [83] associated each input field of a web form with a regular expression that defines valid-input constraints for the input field. It then applied perturbation on regular expressions to generate invalid test inputs. Ref. [94] used DOM models to generate test cases.

5.7. RQ 1.7 – Techniques to derive test artifacts

Fig. 13 shows the distribution of the type of testing/analysis techniques used in the papers. It is natural that this attribute is related to a great extent to results of RQ 1.6 (i.e., source of information to derive test artifacts) with results shown in Fig. 12. The reason is that since whatever source of information is used, an appropriate technique (or techniques) is (are) needed to derive the necessary test artifacts.

In general, majority of the papers used static code analysis (21 papers, 26.6%) and model-based approaches (19 papers, 24.1%) to derive test artifacts. This is also the case in Fig. 12 (source of information) as source code, requirements, logs and models were the most frequent sources of information.

Crawling and requirement-based were also popular and used in 10 papers (12.7%) and nine papers (11.4%), respectively. The remaining types of techniques used were, in order of usage: (1) dynamic code analysis (e.g,. code coverage), (2) model checking, (3) record and playback, (4) concolic testing, (5) symbolic execution, and (6) search-based techniques.

"Other" techniques in this attribute included: classification of user logs [111], agent-based [104], mutation testing [103], anti-random testing [123], and statistical/probabilistic [81] testing.

For papers using model-based techniques to derive test cases, a type of model was to be either reverse engineered for the System Under Test (SUT) or be provided by users. These two types were referred to as "inferred model (automatic)" and "inferred model (manual)" in the previous attribute (Fig. 12).

We noticed that some papers were using more than one technique to derive test artifacts. We thus mapped each paper to as many types of techniques it was using. Details can be found in our online repository and mapping [26]. Fig. 14 depicts the histogram of the number of techniques used in each paper. Fifty-two papers (65.82%) used one technique only. Twenty and four papers (25.31% and 8.86%) used two and three techniques, respectively.

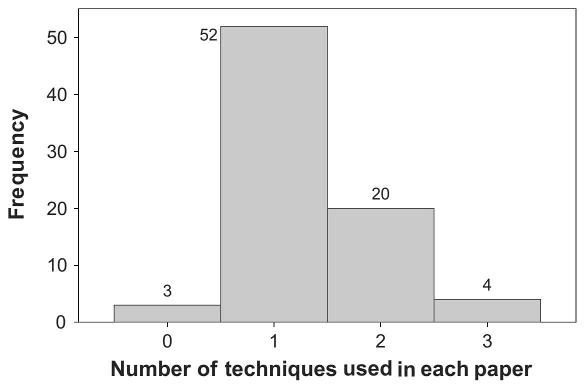


Fig. 14. Number of techniques used in each paper.

For example, Ref. [92] used three techniques (requirements based, search-based, and model-based testing) to compare web testing techniques applied to Ajax web applications in a case study context. Three papers [129,63,91] did not use any particular technique as they were either overview papers [129], or focusing on test processes only [63], or reporting an empirical validation of a web fault taxonomy [91].

5.8. RQ 1.8 – Type of test artifact generated

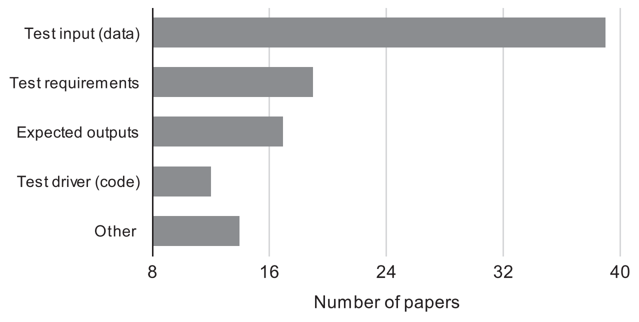
The distribution of generated test artifacts is shown in Fig. 15. The majority (49.4%) of the primary studies created (concrete) test inputs, i.e., input data or input event sequence, in their proposed approach.

The second most preferred test artifact generated was test requirements (27.8%). Recall from Section 4 that test requirements are not actual test input values, but the conditions that can be used to generate test inputs. For example, a coverage-based test technique would generate as test requirement the need to cover a certain control-flow path of a function, i.e., the set of logical expressions that should be made true to cover that control-flow path. By having test requirements, one can manually or automatically generate test inputs, an area of research referred to as test-data generation [41].

Merely 17 papers (21.5%) focused on creating expected outputs to address the oracle problem. Generating automated test drivers (code) received the least attention (15.2%). Although current testing frameworks have built-in test drivers to set up and exercise the system/unit under test, not many proposed techniques have benefited from these frameworks to generate automated test code. Examples of papers that did leverage testing frameworks are [97,57,122,67], which respectively generate test code in JUnit, Fitnesse, Testing and Test Control Notation (TTCN), and Selenium.

Forteen papers (17.7%) generated some other type of test artifacts. For example, Ref. [78] generated a list of interface invocations that do not match any accepted interface. Ref. [96] generated two types of web navigation graphs that are isomorphically compared to spot cross-browser defects. Ref. [55] generated testability measures. Ref. [112] generated reduced test suites and [110] generated test case orders.

In addition, we noticed that some papers generated more than one type of test artifacts. We thus mapped each paper to as many types of test artifacts as it generated. Fifty-four, 18, and four papers (68.35%, 22.78%, and 5.06%) generated one, two, and three types of artifact, respectively. Three papers [116,91,63] generated no test artifacts according to our mapping: [116] presented a symbolic execution framework for JavaScript, a method which can help testing, but is not a testing activity itself. Ref. [91] is an empirical validation of a web fault taxonomy. Ref. [63] evaluated testing processes of web-portal applications and had no need to generate test artifacts. For the details, we refer the interested reader to our online repository and mapping spreadsheet [26].



Test input (data): [125, 92, 111, 120, 87, 121, 106, 114, 96, 54, 100, 88, 115, 130, 123, 93, 101, 128, 62, 67, 75, 70, 82, 99, 61, 76, 109, 56, 59, 95, 126, 83, 90, 108, 89, 86, 58, 64, 65]
Test requirements: [60, 104, 106, 103, 53, 105, 88, 85, 77, 70, 81, 80, 90, 73, 74, 79, 94, 113, 124]
Expected outputs (oracle): [96, 119, 68, 105, 115, 130, 102, 72, 62, 75, 97, 66, 82, 71, 90, 86, 118]
Test driver (code): [114, 130, 122, 67, 97, 71, 109, 95, 86, 64, 57, 129]
Other: [96, 78, 54, 107, 117, 101, 55, 112, 69, 76, 110, 98, 84, 127]

Fig. 15. Type of test artifacts generated.

5.9. RQ 1.9 – Manual versus automated testing

Test automation usually reduces testing effort and therefore it is a quite popular research topic. Forty-nine papers (62.0%) provided full automation for the test approaches they were presenting. For example, [96] presented a fully automated approach for cross-browser compatibility testing.

The techniques in seven papers (8.9%) were fully manual. For example, [125] presented a 2-layer model (based on control flow graphs) for the white-box testing of web applications, in which the model had to be manually provided by the user (tester).

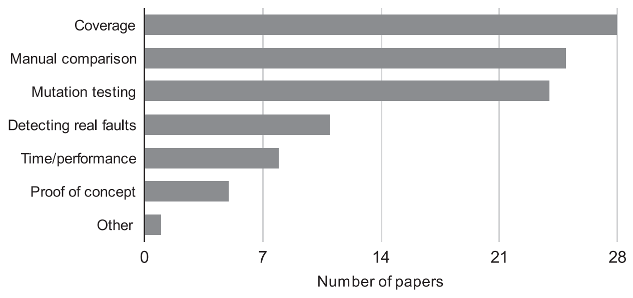
In another set of 20 papers (25.3%), there were both manual and automated aspects (i.e., they were semi-automated). Note that the above three categories add to 76 (=49 + 7 + 20). The remaining three papers in the pool [91,63,69] were not directly involved with test techniques, but with other aspects of testing web applications. For example, for [91] which presented an empirical validation of a web fault taxonomy, manual or automated testing was not applicable.

5.10. RQ 1.10 – type of the Evaluation Method

We looked for methods used for evaluating proposed testing techniques. Our mapping includes papers that use coverage criteria (e.g., code coverage or coverage based on a model), fault injection (a.k.a. mutation testing), and manual comparison. The distribution of these evaluation methods is depicted in Fig. 16. Coverage criteria (used in 25 papers, 31.65%) and manual comparison (used in 22 papers, 27.85%) were the most preferred approaches. Mutation testing was used in 14 papers (17.72%). Thirty-six papers (45.57%) used other evaluation methods to validate their proposed testing technique. For example, [81,79,124] each conducted a type of reliability analysis as its evaluation method. Ref. [113] used test suite size as its evaluation metric.

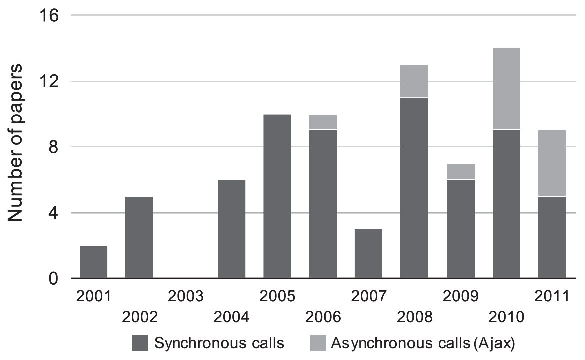
5.11. RQ 1.11 – Static web sites versus dynamic web applications

A majority of papers (75 papers, 94.9%) investigated dynamic web applications and only four (5.1%) of the papers were geared towards static web sites.



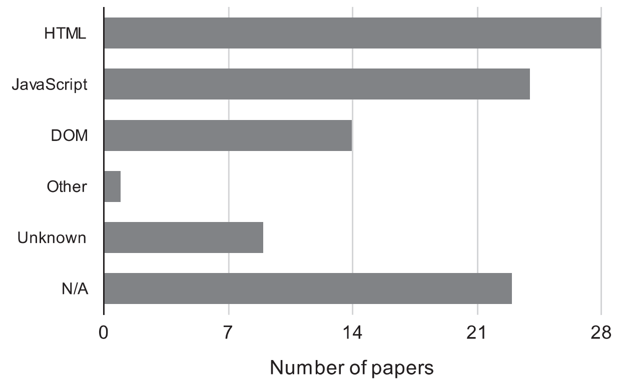
Coverage (code, model): [125, 60, 111, 120, 116, 87, 104, 121, 106, 114, 53, 101, 102, 62, 77, 112, 70, 99, 76, 56, 90, 74, 108, 89, 86, 64, 94, 113]
Mutation testing (fault injection): [92, 111, 121, 114, 103, 68, 88, 115, 102, 91, 112, 70, 80, 110, 56, 95, 83, 90, 73, 74, 57, 94, 113, 118]
Manual comparison: [106, 114, 96, 78, 119, 54, 107, 93, 101, 77, 82, 99, 80, 61, 76, 109, 59, 98, 131, 126, 83, 89, 58, 127, 129]
Time/performance: [111, 78, 119, 105, 102, 97, 76, 131]
Detecting real faults: [100, 88, 128, 62, 97, 71, 69, 99, 80, 61, 83]
Example (proof of concept): [125, 85, 117, 122, 66]
Other: [111, 54, 77, 81, 109, 131, 79, 113, 124]

Fig. 16. Type of evaluation method.



Synchronous calls: [125, 111, 120, 116, 87, 104, 121, 106, 114, 103, 78, 119, 54, 53, 68, 105, 100, 88, 115, 130, 107, 123, 93, 85, 117, 101, 128, 72, 91, 63, 62, 122, 67, 75, 77, 55, 112, 70, 66, 81, 82, 71, 99, 80, 61, 76, 110, 59, 98, 126, 83, 90, 74, 108, 89, 86, 58, 64, 57, 79, 84, 65, 127, 124, 118, 129]
Asynchronous calls (Ajax): [92, 60, 96, 102, 96, 69, 109, 56, 95, 131, 73, 94, 113]

Fig. 17. (A) synchronicity of HTTP calls (annual trend).



HTML: [125, 92, 60, 106, 103, 119, 100, 88, 130, 123, 93, 85, 91, 122, 75, 55, 81, 82, 69, 99, 83, 90, 86, 64, 79, 65, 124, 118]
JavaScript: [125, 92, 60, 116, 104, 96, 100, 130, 107, 123, 93, 102, 91, 55, 97, 70, 71, 80, 109, 95, 73, 94, 65]
DOM: [92, 96, 100, 102, 75, 97, 80, 109, 56, 95, 83, 90, 86, 94]
Other: [123, 76, 56]
Unknown: [53, 67, 66, 82, 59, 131, 108, 58, 113]
N/A: [111, 120, 87, 121, 114, 78, 54, 105, 115, 117, 101, 128, 72, 62, 77, 112, 61, 110, 98, 126, 89, 57, 84]

Fig. 18. Client-tier web technologies used.

It is clear that testing dynamic web applications is a more popular area (perhaps due to being clearly more challenging) compared to testing static web sites. The handful number of papers in testing static web sites are [81,124,79,82] which has been published in 2001 [81], 2006 [124,79], and 2008 [82]. For example [81] proposed an approach for measuring and modeling the reliability of static web sites using statistical testing. The technique extracts the web usage and failure information from existing web logs. The usage information is then used to build models for statistical web testing. The related failure information is used to measure the reliability of Web applications and the potential effectiveness of statistical web testing.

We can clearly observe that with the introduction of dynamic web languages and technologies in the early 2000s, e.g., JSP and PHP, researchers have focused on testing dynamic web applications, rather that static web sites. This is since the former type of applications have much more chances of having defects compared to the latte. In static web sites, defects are usually quite of limited types, e.g., broken links and invalid resources (e.g., images), which can be detected quite easily using a large collection of automated tools (e.g., LinkChecker).

5.12. RQ 1.12 – Synchronicity of HTTP calls

In terms of synchronicity of HTTP calls, 66 papers (83.5%) have targeted synchronous HTTP calls, while only 13 papers (16.5%) have targeted asynchronous (Ajax-based) calls.

Although asynchronous calls make web applications more complex, and web development more error-prone, the number of papers on this topic is much lower than on synchronous calls. One of the reasons is that asynchronous client–server communication has been a relatively recent technology through the adoption of the `XMLHttpRequest` object in modern browsers. Ajax, a technique advocating the use of asynchronous server calls, was first introduced in 2005 [28].

As a stack bar chart, Fig. 17 shows the annual trends of the focus on synchronous web calls versus asynchronous web calls (Ajax). It is clear from the two trends that testing Ajax-based web applications is starting to attract more researchers in recent years, and its general trend is in increasing slope.

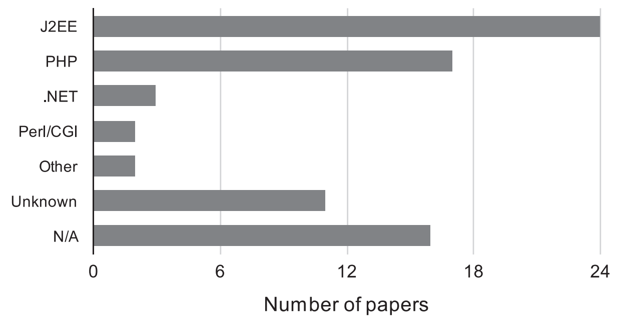
5.13. RQ 1.13 – Client-tier web technologies

Recall from Section 5.4 (RQ 1.4: test locations) that, respectively, 22, 41 and 14 papers in the pool focused on client-side testing, server-side testing, or both locations. Fig. 18 shows the distribution of the client-tier web technologies used.

HTML, Document Object Model (DOM), and JavaScript are the three top main technologies in this context, appearing in 28, 14, and 24 papers (35.44%, 17.72%, and 30.38%), respectively. One paper (1.27%) [123] was categorized under "other" client-tier web technologies, i.e., it discussed the testing implications of browser cookies.

For 23 papers (29.1% of the pool), the client-tier web technology was 'not applicable' (shown as N/A in Fig. 18). This was due to the fact that either the paper under study was focused on server-side application testing only (e.g., [110]), or the paper was not discussing test techniques, per se. For example, [84] titled 'Verifying Interactive Web Programs', is a paper in conceptual level, i.e., it presented formalisms such as control-flow graph for web applications and the technique seemed to be quite neutral of any client- or serve-tier web technology.

As per our analysis, nine other papers (11.4%), which were on client-tier testing, did not explicitly discuss the client-tier technologies used. We also did not intend to judge the technologies based on our own interpretations. These papers are shown as "Unknown" in Fig. 18.



J2EE (Servlet, JSP, etc.): [111, 120, 104, 121, 114, 103, 78, 119, 100, 88, 115, 130, 85, 112, 99, 76, 110, 95, 131, 74, 86, 57, 94, 113]
PHP: [125, 87, 121, 53, 68, 107, 93, 128, 72, 91, 62, 75, 61, 98, 131, 90, 127]
.NET: [101, 91, 89]
Perl (CGI): [105, 70]
Other: [100, 129]
Unknown: [54, 67, 66, 71, 59, 108, 58, 64, 79, 84, 124]
N/A: [60, 116, 106, 96, 117, 102, 81, 82, 69, 80, 109, 56, 83, 73, 65, 118]

Fig. 19. Server-tier web technologies used.

5.14. RQ 1.14 – Server-tier web technologies

Fig. 19 shows the frequency of server-tier web technologies used in the papers.

J2EE, PHP, .NET, Perl/CGI were the most focused-on server technologies appearing in 24, 17, three, and two papers (30.38%, 21.52%, 3.80%, and 2.53%), respectively.

In Fig. 18, the bar labeled as "Other" includes papers targeting technologies such as XML, Lisp, PLT scheme, and Portlet.

As per our analysis, we decided that 11 papers (13.9%) were concerned with server-tier testing, but did not explicitly discuss the server-tier technologies they were focusing on.

After having studied client-tier and server-tier web technologies in isolation, we found that it would be useful to also conduct a cross-analysis of number of studies in different client-tier versus server-tier web technologies. Fig. 20 depicts this information. Recall from the above discussions that in some papers, the client-tier was not applicable (N/A) since they were fully focusing on server-tier testing and, likewise, in some papers, server-tier web technologies were not present. As we can see in this figure, there are a handful number of techniques targeting both client and server side.

5.15. RQ 1.15 – Tools presented in the papers

Forty-one papers (51.9% of the pool) presented at least a tool, which were mostly of research-prototype strength. We believe this is quite a positive sign for the web testing community as about half

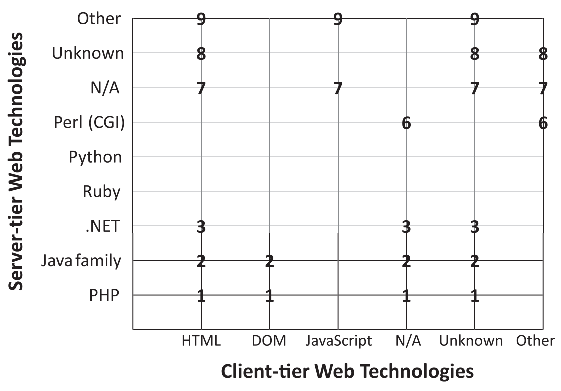


Fig. 20. Cross analysis of number of studies in different client-tier versus server-tier web technologies.

of the papers in the pool provided tool support (automation) along with proposing test techniques.

As a randomly-chosen set of names, the following tools were presented: Artemis [60], Kudzu [116], TestWeb [106], webMuJava [103], and CrossT [96].

We thought that a natural question to ask in this context is whether the presented tools are available for download, so that other researchers or practitioners could use them as well. We only counted a presented tool available for download if it was explicitly mentioned in the paper explicitly. If the authors had not explicitly mentioned that the tool is available for download, we did not conduct internet searches for the tool names. The result was somewhat surprising. Only six of the 41 tool-presenting papers explicitly mentioned that their tools are available for download.

5.16. RQ 1.16 – Attributes of the web software SUT(s)

As discussed in Section 4 (Classification Scheme), and shown in Table 2, we extracted the following attributes for the web software SUT(s), discussed in each paper:

1. Number of SUTs used in each paper.
2. The name(s) of the SUT(s).
3. Total LOC size of the SUT(s).
4. For each SUT, its scale which could be a type in this set {Academic experimental, Open-source, Commercial}.
5. Other size metrics (e.g., # of pages, # of form).

Number of SUTs used in each paper: We discuss next the data and findings for the above items #1 through #5.

5.16.1. Number of SUTs used in each paper

Fig. 21 depicts the histogram of the number of SUTs used in each paper. 16, 23 and nine papers (20.25%, 29.11%, and 11.39%) evaluated their approaches on 1, 2, and 3 SUTs, respectively. There were six papers (7.59%), each using more than 10 SUTs. The paper with the highest number of SUTs (53 of them) was [80], published in 2011. The average number of SUTs per papers was 4.8.

By statistically reviewing the number of SUTs analyzed in each paper, we hypothesized that there might be correlation between number of SUTs analyzed in each paper, and its research facet type (e.g., solution proposal, validation research, and evaluation research). To systematically analyze this hypothesis, we cluster the papers by their research facet types. Fig. 22 shows the individual-plot of number of SUTs for each of the three research-facet-type clusters. As we can see, the three distributions are mostly overlapping, thus rejecting the above hypothesis. Thus, it is not necessarily true that papers with more mature research facet types would have applied their methods to more number of SUTs.

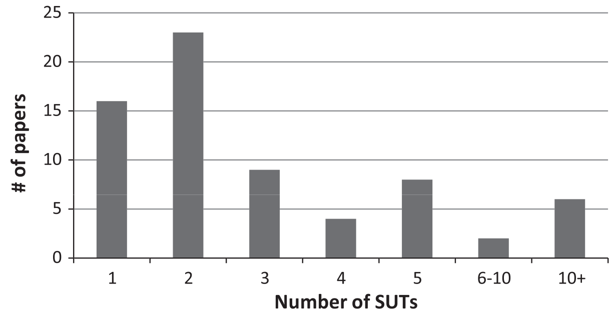


Fig. 21. Number of SUTs analyzed in each paper.

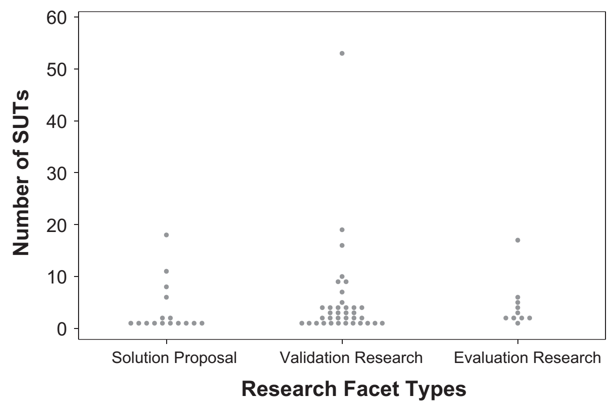


Fig. 22. Cross analysis of number of studies in different client-tier versus server-tier web technologies.

5.16.2. Names of the SUTs

We recorded the name(s) of the SUT(s) used in each paper and conducted some brief statistical/text analysis on the list of names. In total, 176 unique SUT names were used, while the sum of the number of SUTs used in all the papers was 210. This indicated that some SUTs were used in more than one paper. For example, we found that an open-source e-commerce bookstore web application [1] has been used for evaluations in eleven papers.

The followings are also examples of the real web applications used as SUTs for testing in the papers: (1) Yahoo Auto Classifieds (used in [65]), (2) http://www.msn.com (used in [131]), and (3) Facebook Chat (used in [116]).

Listing the names of all 176 different SUTs in this article is impossible, but can be found in our online repository [26].

5.16.3. LOC size of the SUTs

Only 41 of the 79 primary studies reported the LOC size of their SUTs. Fig. 23 depicts the histogram of the total LOC size of the SUT(s) used in each paper. It is good to see that many papers have used relatively large SUTs (more than one or even 10 KLOC) for evaluating their approaches. The paper with the largest SUT LOC sizes was [72], published in 2011, in which only one large-scale SUT (named Mambo) was used, having a total of 601 KLOC. The average number of SUTs per papers was 75, 610.35.

In this context, we hypothesized that the LOC of SUTs may have been increasing in newer papers. To visually assess this hypothesis, Fig. 24 depicts as an *X*–*Y* (scatter) plot the LOC size of SUT(s) used in each paper versus year of publication of each paper. Each dot in this plot corresponds to a single paper. Note that the *Y*-axis in this figure is in logarithmic scale. The correlation value of this data set if only 0.33, meaning that there is only a weak correlation between the year of publication and SUT size. It is still nice to observe that larger and larger SUT sizes appear in more recent papers. The three papers [72,68,96] rank, in order, 1st, 2nd and 3rd in terms of having the highest LOC sizes of SUTs.

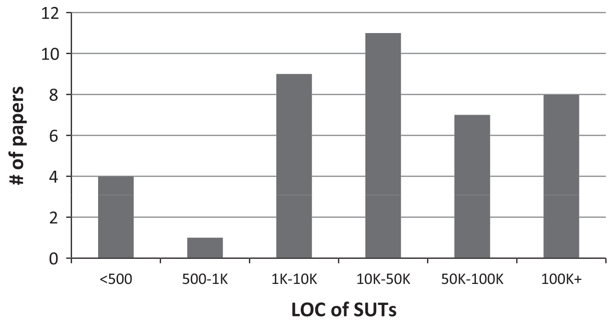


Fig. 23. SUT sizes in LOC.

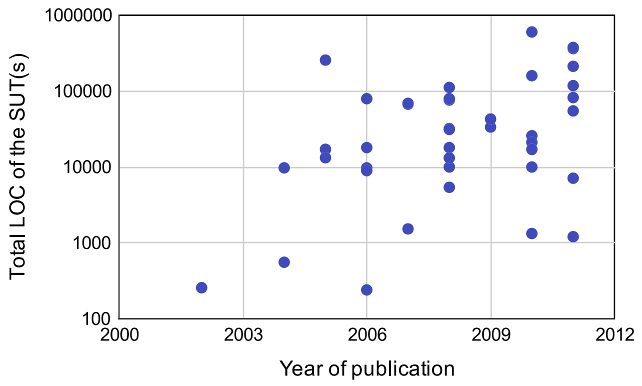


Fig. 24. SUT size (LOC) versus publication year.

It should be noted in this context that most of the papers have not applied their testing technique to the entirety of each SUT that they selected, but only to a or few selected sub-system(s) of each SUT. Thus, the LOC sizes of the SUTs are not entirely precise in terms of the scale of the system and evaluation conducted in each paper.

5.16.4. Types of SUTs: academic experimental, real open-source, or commercial software

In addition to LOC, we wanted to assess the frequency of using academic experimental, real (not academic) open-source, or commercial SUTs in the papers. Fig. 25 depicts the histogram of that information. Forty-two papers (53.16%) used open-source SUTs, 22 papers (27.85%) used academic experimental systems, and 21 papers (26.58%) used commercial applications. A brief conclusion from observing these numbers is that relatively a small number of academic experimental web systems have been used (i.e., tested) in the papers in contrast to the large number of open-source software.

5.16.5. Other size metrics

In addition to LOC measures and statistics, some studies reported other size metrics for their SUTs, e.g., # of pages, # of sessions, # of forms, # of methods, and # of classes. We also recorded them in our online data repository [26] whenever we noticed such reported data.

Forty-six papers reported other size metrics, which denotes, as a good indicator, attention to detail and high level of transparency in the web testing literature. For example, the authors of [126] reported that they applied their statistical testing method to 2046 sessions of their SUT.

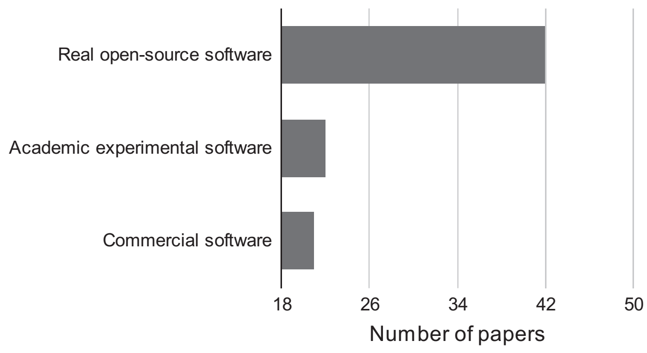


Fig. 25. Type of web software as SUTs.

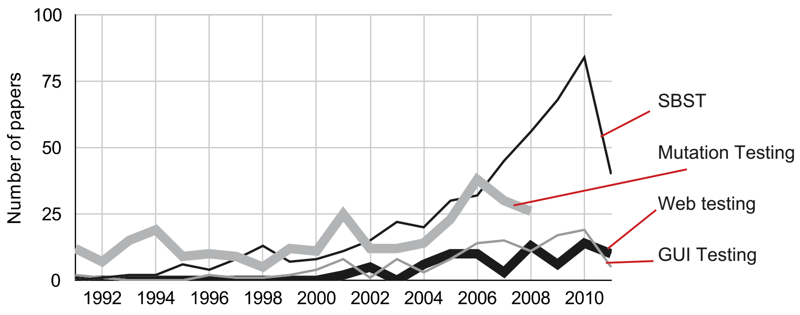


Fig. 26. Publication trend per year.

6. Demographic trends and bibliometrics

To address RQ 2, we analyzed the demographic trends and bibliometrics of the web testing literature. We will elaborate on the results for each related sub question (RQ 2.1–RQ 2.5) in the following subsections.

6.1. RQ 2.1 – publication trend per year

The annual publication volume of web testing papers is shown in Fig. 26. To conduct cross-area comparisons, we compare the trend of web testing publications to the publication trends of three other software testing areas: Graphical User Interface (GUI) testing, search-based software testing (SBST), and mutation testing. Data for these three other subjects have been extracted from [11,52,33,34], respectively.

In terms of starting year of the publication range, we can see that web testing papers started to appear in 2001, however, the other three areas have a longer history, especially mutation testing. This is probably due to the fact that the web technology has a relatively younger history compared to the general software engineering research.

Compared to the other three, the publication volume trend for web testing is relatively less stable, e.g., our pool had 0 papers from the year 2003. In terms of the absolute number of papers, according to the above data sources, as of this writing, web testing, GUI Testing, SBST, and mutation testing domains have 79, 122, 474, and 289 papers, respectively.

Of course, we should note that there are overlaps between the different areas. For example, a paper can present a web search-based method, e.g., [94], or another paper in our pool uses the *Selenium* web GUI test tool [15], which falls under both web and GUI testing domains.

6.2. RQ 2.2 – Citation analysis and top-cited papers

This RQ intended to identify the top-cited papers. Since the papers under study were published in different periods, we deemed it appropriate to consider the publication year of each paper in analyzing its citation count. With citation numbers constantly increasing, we should note that citation data used for our analysis were extracted from Google Scholar on April 7, 2012. For papers that have appeared both as a conference publication and a journal extension (e.g., [70]), we count the citations to both versions.

Fig. 27 visualizes the citation count of each paper vs. publication year as an *X*–*Y* plot. Two types of metrics are shown in this figure: (1) absolute (total) value of number of citations, and (2) normalized number of citations which is defined as follows:

$$NormalizedCitations(p) = \frac{TotalCitations(p)}{2012 - PublicationYear(p)}$$

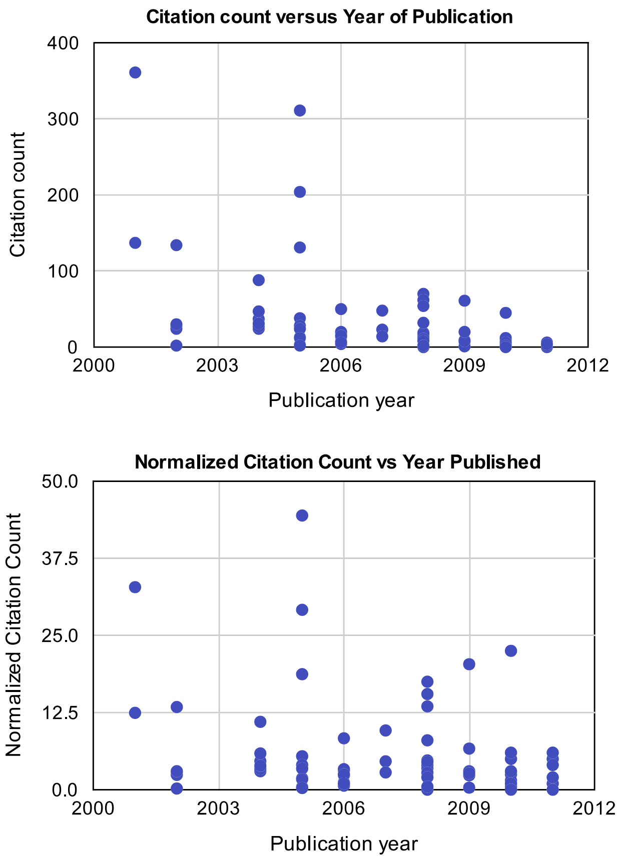


Fig. 27. Citation count vs. publication year.

For example, [106] has 361 total citations as of this writing and was published in 2001. Thus, its normalized citations is calculated as:

$$NormalizedCitations([106]) = \frac{361}{2012 - 2001} = 32.8$$

Compared to the absolute value, the normalized metric essentially returns the average number of citations of a paper per year, since its publication year.

The top three publications with the most absolute and normalized citations include [106,70,58]. Filippo Ricca and Paolo Tonella [106] were among the very first researchers to publish a paper on web testing in 2001. Their ICSE 2001 paper [106] has 361 total citations and 32.8 normalized citations. That paper is seen as one of the flagship papers in the web testing domain, and is thus cited by many researchers in this area.[8] Elbaum et al. [70] were the first to

[8] This paper won an ACM SIGSOFT Most Influential Paper Award in 2011.

propose the use of 'user session data' for web application testing. With 311 citations, their paper (ICSE'03 and TSE'05 extension) has the highest normalized citation count (44.4). Andrews et al. [58] proposed to model web applications as Finite State Machines (FSMs) and generate test cases.

A histogram of the citations, based on the two metrics, for all papers in our pool is shown in Fig. 28. Note the *x*-axes in the two graphs have different scales. Only four papers [101,67,90,73] have had no citations at all. It is easy to see that both distributions are leaning towards the left side, thus indicating that most papers have relatively a small number of citations.

6.3. RQ 2.3 – Most active researchers in the area

To get an overview of active researchers in this area, we follow a similar approach as other bibliometric/ranking studies in software engineering, such as [48,27,30,29]. As the metric, we count the number of papers published by each author. To keep the brevity of the ranking results, we show the order of top authors who have published at least three papers in the pool in Fig. 29.

The competition is close as the second and third ranks are in tie. The ranking is as follows: Paolo Tonella (12 papers), Filippo Ricca and Lori Pollock with nine papers, Alessandro Marchetto, Sara Sprenkle and Sreedevi Sampath, each with seven papers.

6.4. RQ 2.4 – Active nations

Similar to other bibliometric studies in software engineering, e.g., [52], we ranked the most active countries based on the affiliation of the authors who have published web testing papers. The rationale for this ranking is to know the researchers of which countries (as a group) focus more on web application testing. Similar studies to rank the most active nations have been done in other areas of science, e.g., [10].

If an author had moved between two or more countries, we attributed each of his/her papers to the explicit affiliation information on top of each paper. If a paper was written by authors from more than one country, we incremented the counters for each of those countries by one.

Fig. 30 shows the ranking of countries from which at least two papers have been published in this area. The top three countries are: (1) USA (with 40 papers, 50.6%), (2) Italy (with 18 papers, 22.8%), and Canada (nine papers, 11.4%).

We had access to the country breakdown of authors from another recent mapping study in testing (namely, GUI testing) [11]. Thus, as another analysis, it would be interesting to compare the two datasets and assess the activity of the top nations in these two related areas of software testing. The country listing for the domain of GUI testing is taken from [11] and is shown in Fig. 31. Note that the paper pool in the mapping study done in [11] is 122, which denotes that the GUI testing literature is larger (in terms of paper quantity) than web testing.

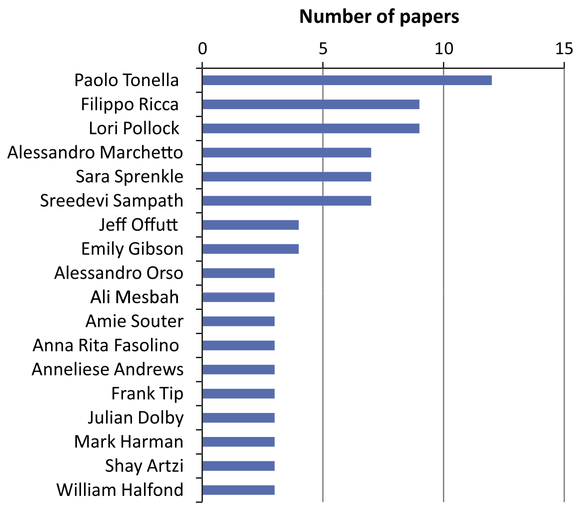


Fig. 29. Authors with at least three related papers (2001–2011).

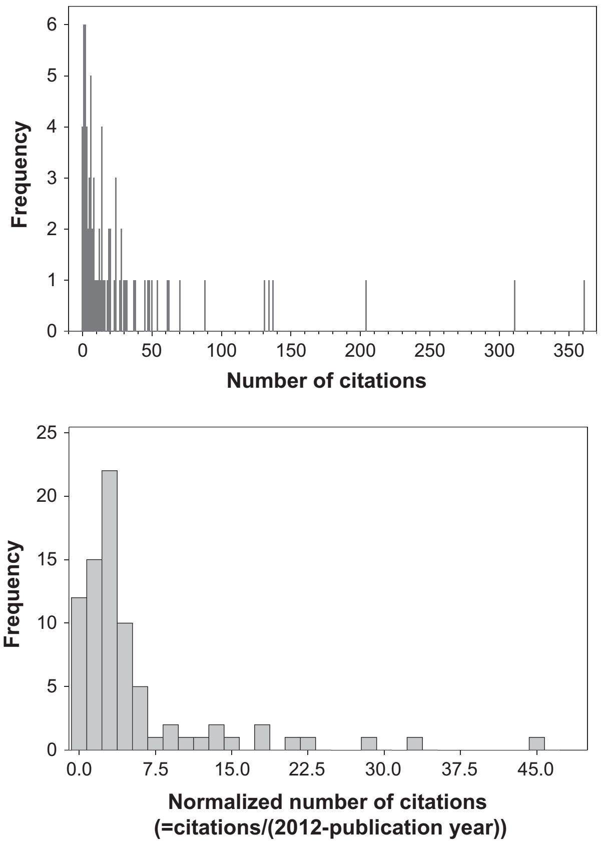


Fig. 28. Histogram of number of citations for all papers included in our study.

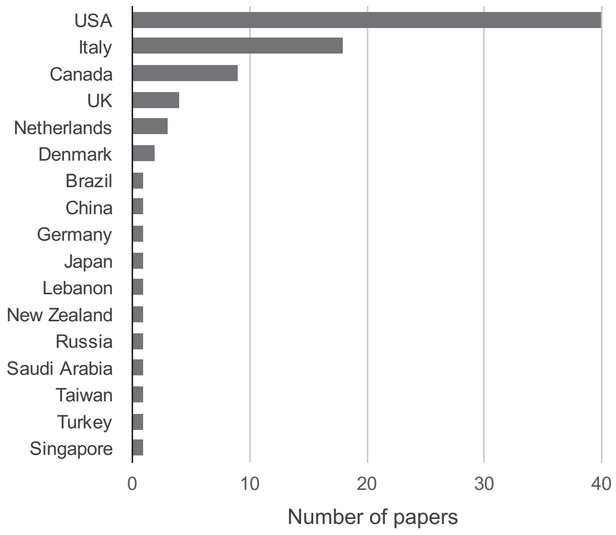


Fig. 30. Countries contributing to the web testing literature (based on author affiliations).

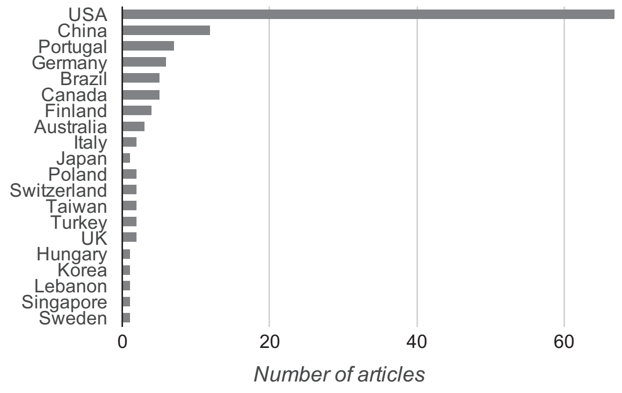


Fig. 31. Countries contributing to GUI testing literature. *Source*: [11].

There are both similarities and differences between the trends in Figs. 30 and 31. Both domains (web and GUI testing) is dominated by the US researchers, 50.6% and 54.9%, respectively.

To evaluate the degree of contribution to the web testing literature across the world, we could easily calculate the internationalization index in this context, i.e., the number of countries represented in these papers divided by the number of nations in the world. We considered the value of 193 for the number of nations, as the United Nations has 193 member states. There are 19 and 20 countries represented in web and GUI testing, respectively and the respective internationalization indices are 9.85% and 1.36%. This denotes that the two areas have contributions from only 10 of the world countries.

According to our analysis, 71 (89%) of the papers have been authored by authors from one nation only, while eight papers (11%) [54,60,71,72,83,93,96,102] have been written by authors from more than one country.

The followings are the list of internationally-authored papers and the collaborating nations. As we can see, collaborations between researchers from USA and Canada are the highest in this category. This information provides a high-level snapshot of the level of international collaborations in the web testing community, and could be compared to other software engineering/testing subfields once their data is available.

- USA, Canada [102,72,96].
- USA, Denmark [60].
- USA, China [83].
- USA, UK [71].
- Italy, UK [93].
- Saudi Arabia, UK [54].

6.5. RQ 2.5 – Top venues

To rank the venues, we used two metrics: (1) the number of papers published in each venue and (2) the total number of citations to web testing papers published in each venue. Using the first metric, the ranking of the top venues with at least three papers is shown in Table 3. There are 13 venues in this list: 10 conferences/symposia, and three journals. Many major software engineering conferences and journals are in this list. For example, the venue with most papers (six papers, 7.59%) is ICST, which has recently attracted and published many web-testing related papers.

We then measured the total number of citations to web testing papers published in each venue (as we had in our paper pool). Table 4 shows the top 10 venues ranked by number of citations. Expectedly, many of the venues in the two Tables 3 and 4 are overlapping. According to the values in Table 4, ICSE and TSE (seen by many software engineering researchers as the top two software engineering venues) have the highest ratio of citations to number of papers.

Table 3
Venues with at least three papers, ranked by number of papers.

Venue	Acronym	#
Int. Conf. on Software Testing	ICST	6
Int. Conf. on Automated Software Engineering	ASE	5
Information and Software Technology	IST	5
Int. Conf. on Software Engineering	ICSE	5
Int. Conf. on Web Engineering	ICWE	4
Int. Symposium on Software Reliability Engineering	ISSRE	4
Int. Symposium on Software Testing and Analysis	ISSTA	4
Int. Symposium on Web System Evolution	WSE	4
Transactions on Software Engineering	TSE	3
Int. Symposium on Foundations of Software Engineering	FSE	3
Int. Conf. on Software Maintenance	ICSM	3
Software Tools for Technology Transfer	STTT	3
Int. Conf. on World Wide Web	WWW	3

Table 4
Top 10 venues ranked by number of citations.

Venue	# Of papers	# Of citations
TSE	3	498
ICSE	5	438
SoSym	2	209
ICSM	3	158
ISSTA	4	155
WWW	3	135
ISSRE	4	131
ASE	5	126
ICST	6	122
FSE	3	64

7. Discussions

Based on the research work-flow of this study as presented in Fig. 1, summarized discussions and implications of this study along with some of the threats to validity are presented in this section.

7.1. Findings, trends, and implications

First of all, we witness a clear increase in the number of research papers in the area of web application testing.

Most of the published papers propose new testing techniques or an adoption of existing software testing techniques geared towards the web domain. The most popular techniques used for testing web applications include static analysis, model-based testing, and crawling. Analysis and testing techniques such as fault-based testing (e.g., mutation testing), symbolic execution, concolic testing, and search-based have gained very limited attention from this community so far.

Our answer to RQ 1.1 indicates that about half of the papers mention an accompanying tool implementation in their paper. However, merely a few papers (6/79, or 7.59%) have a tool that can be downloaded (RQ 1.15). This is an alarming fact that needs to be taken more seriously by the web testing research community, especially if they intend to have an impact in industry.

As far as the evaluation methods are concerned (RQ 1.2), the majority of the papers present an empirical validation of their proposed solutions. However, most of these validations are conducted on self-implemented or small open-source web applications. We did not encounter many “experience” papers in our study. Conducting empirical evaluations on large-scale industrial web applications is an area that still needs much attention from this research community.

The primary focus of the papers seems to be on automation and coverage (code or model), while (automating) the oracle problem has received relatively limited attention (RQ 1.3).There is relatively much less work on "exploratory" web testing (test activity being based on human knowledge), which is not in line with the regular pervasive practice of manual ad hoc testing of web applications in the industry.

The majority of the work has focused on testing the server-side in the past (RQ 1.4). However, in order to create more responsive modern web applications, much of the web application state is being offloaded to the client-site. That is why we see an increasing trend towards papers that target client-side testing and we believe this trend will continue in the coming years. RQ 1.5 revealed the need for more integration testing in the web context.

RQ 1.6 revealed that white-box testing is the most popular approach to derive test artifacts in the area, while new approaches such as those based on invariants have been proposed in recent years.

RQ 1.7 showed that a variety of various techniques have been used and proposed to derive test artifacts, such as search-based testing. Even, there are papers using up to three different techniques together.

In term of type of test artifact generated, RQ 1.8 revealed that while many paper generate test data in abstract notion, few papers propose approaches to automatically create test driver (automated test code). There have been a mix of manual and automated testing (RQ 1.9).

Various types of evaluation methods have been used (RQ 1.10), e.g., coverage and mutation testing.

Most papers focused on testing dynamic web applications, rather than static web applications (RQ 1.11).

With the introduction of Ajax and similar technologies in recent years, the community is gradually shifting its focus on applications with asynchronous (Ajax-based) calls (RQ 1.12).

On the client-side (RQ 1.13), HTML has received most of the attention so far. JavaScript, a dynamic loosely-typed language that is known to be challenging to test, is getting increasingly more attention. The dynamic DOM, which plays a prominent role in the realization of modern web applications, will need more spotlight. Surprisingly, there is no work on CSS, a widely used language for defining the presentation semantics of web pages.

On the server-side (RQ 1.14), however, most of the research work has revolved around J2EE and PHP. There is limited work on server-side languages such as Python, Ruby, and almost none on the nowadays popular Node.js framework (Server-side JavaScript).

SUT's with various (LOC) sizes and either open-source or commercial systems have been the subject of evaluations in different papers (RQ 1.16).

An interesting observation we have made is that almost every paper is using a different set of web applications (SUTs) for their validations, which makes tool or technique comparisons quite challenging in this field. Having a set of subject systems that everyone can use for rigorous controlled experimentation is needed here. Ideally, researches in this field should create a web application-artifact repository, similar to the Software-artifact Infrastructure Repository,[9] which hosts many Java and C software systems, in multiple versions, together with supporting artifacts such as test suites, fault data, and scripts.

[9] http://sir.unl.edu/portal/.

7.2. Threats to validity

The results of a systematic mapping study 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.

7.2.1. Internal validity

We presented in Section 3.2 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 are discussed in Section 3.2.2. The decision on which papers to include in the final pool depended on the group judgement of the researchers conducting the systematic mapping study. 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 systematic mapping study 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.

7.2.2. Construct validity

Construct validity is concerned with the extent to which 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 in Section 4, once the classification scheme was developed, the papers in the pool were sorted into the scheme, i.e., the actual data extraction took place. The pool of papers were partitioned to the authors (i.e., each author was assigned about 27 papers). Each author first extracted the data by mapping the paper inside the classification scheme independently. Then a systematic peer review process 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.

We should mention that for tool availability, we relied on the information in the papers. If the authors did not explicitly mention that their tool was available for download, we did not conduct internet searches for the tool. Furthermore, recall from 5.16 (RQ 1.16: attributes of the SUTs) that most of the papers have not applied their testing technique to the entirety of each SUT that they selected, but only to a or few selected sub-system(s) of each SUT. Thus, the LOC sizes of the SUTs are not entirely precise in terms of the scale of the systems and evaluations conducted in each paper.

7.2.3. Conclusion validity

It is important for a systematic mapping 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 Sections 5, 6 and 7.1 graphs generated directly from the data and discussed the explicit observations and trends. This ensures a high degree of traceability between the data and conclusions. In addition, our mapping data are available in the form of an online repository, for others to validate.

7.2.4. External validity

The results of the systematic mapping study were considered with respect to approaches in the software engineering domain. Thus, the classification map and data presented and the conclusions drawn are only valid in the given context (web application testing). The classification scheme presented in this paper can serve as a starting point for future studies. Additional papers and approaches that are identified in the future can be categorized accordingly. Due to the systematic procedure followed during the mapping study, we believe our study is repeatable.

8. Conclusions and future work

The web has proven to be a powerful medium for delivering software services over the Internet. Due to its inherit 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 present a first systematic mapping of the papers in the area of web application functional testing, published between 2000 and 2011. Our initial search retrieved 147 papers of which 79 were included in this study using a selection strategy. We incrementally derived a classification scheme by analyzing the included papers and used that scheme to conduct the mapping.

In addition, we present a first bibliometrics analysis of the domain to gain an understanding of the publication trend per year, citations, active researchers and venues in the area.

Our study indicates that web testing is an active area of research with an increasing number of publications. Our mapping 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 spotting research areas that need more attention. For instance, areas that need additional investigation for web application testing include automated oracle generation, mutation testing, concolic testing, testing asynchronous client/server interactions, coverage metrics (e.g., state coverage and code coverage), test support for server-side languages such as Ruby and Python, and client-side DOM and CSS.

As future work, based on this study, we intend to conduct a systematic literature review of the field to analyze the existing evidence for different web testing techniques and their effectiveness. Also, an interesting additional classification would be related to the domain of the web application under test (e.g., medical, financial, academic domains).

Acknowledgements

Vahid Garousi was supported by the Discovery Grant #341511-07 from the Natural Sciences and Engineering Research Council of Canada (NSERC), and also by the Visiting Scientist Fellowship Program (#2221) of the Scientific and Technological Research Council of Turkey (TÜBITAK). Ali Mesbah was supported by NSERC through its Discovery and Strategic Project Grants programmes.

References

[1] An Open Source E-Commerce Bookstore (book), Open Source Web Applications with Source Code <http://www.gotocode.com>.
[2] Context-Driven School of Testing <http://www.context-driven-testing.com> (accessed April 2012).
[3] W. Afzal, R. Torkar, R. Feldt, A systematic mapping study on non-functional search-based software testing, in: 20th International Conference on Software Engineering and Knowledge Engineering (SEKE 2008), 2008.
[4] W. Afzal, R. Torkar, R. Feldt, A systematic review of search-based testing for non-functional system properties, Information and Software Technology 51 (2009) 957–976.
[5] M.H. Alalfi, J.R. Cordy, T.R. Dean, Modelling methods for web application verification and testing: state of the art, Software Testing, Verification and Reliability 19 (2009) 265–296.
[6] S. Ali, L.C. Briand, H. Hemmati, R.K. Panesar-Walawege, A systematic review of the application and empirical investigation of search-based test case generation, IEEE Transactions on Software Engineering 36 (6) (2010) 742–762.
[7] S. Ali, L.C. Briand, H. Hemmati, R.K. Panesar-Walawege, A systematic review of the application and empirical investigation of search-based test case generation, IEEE Transactions on Software Engineering 36 (2010) 742–762.
[8] D. Amalfitano, A. Fasolino, P. Tramontana, Techniques and tools for rich internet applications testing, in: Proceedings 12th IEEE International Symposium on Web Systems Evolution (WSE), IEEE Computer Society, 2010, pp. 63–72.
[9] P. Ammann, J. Offutt, Introduction to Software Testing, Cambridge University Press, 2008.
[10] E. Archambault. 30 Years in Science: Secular Movements in Knowledge Creation <http://www.science-metrix.com/30years-Paper.pdf>.
[11] I. Banerjee, B.N. Nguyen, V. Garousi, A.M. Memon, Graphical User Interface (GUI) Testing: Online Repository <http://www.softqual.ucalgary.ca/projects/2012/GUI_SM/> (accessed April 2012).
[12] Z.A. Barmi, A.H. Ebrahimi, R. Feldt, Alignment of requirements specification and testing: A systematic mapping study, in: Proceedings of the 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, ICSTW '11, 2011, pp. 476–485.
[13] R.V. Binder. Testing object-oriented software: a survey, in: Proceedings of the Tools-23: Technology of Object-Oriented Languages and Systems, 1997, p. 374.
[14] M. Bozkurt, Y.H.M. Harman, Testing web services: a survey, in: Technical Report TR-10-01, Department of Computer Science, King's College London, 2010.
[15] C.T. Brown, G. Gheorghiu, J. Huggins, An Introduction to Testing Web Applications with Twill and Selenium, O'Reilly Media, 2007.
[16] D. Budgen, M. Turner, P. Brereton, B. Kitchenham, Using Mapping Studies in Software Engineering, in: Proceedings of PPIG 2008, Lancaster University, 2008, pp. 195–204.
[17] G. Canfora, M.D. Penta, Service-oriented architectures testing: a survey, in: International Summer Schools on Software Engineering, 2008, pp. 78–105.
[18] B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, R. Koschke, A systematic survey of program comprehension through dynamic analysis, IEEE Transactions on Software Engineering 35 (5) (2009) 684–702.
[19] P.A. da Mota Silveira Neto, P. Runeson, I. do Carmo Machado, E.S. de Almeida, S.R. de Lemos Meira, E. Engstrom, Testing software product lines, IEEE Software 28 (2011) 16–20.
[20] P.A. da Mota Silveira Netoa, I. do Carmo Machadoa, J.D. McGregord, E.S. de Almeidaa, ilvio Romero de Lemos Meiraa, A systematic mapping study of software product lines testing, Information and Software Technology 53 (5) (2011) 407423.
[21] G.A. Di Lucca, A.R. Fasolino, Testing web-based applications: the state of the art and future trends, Information and Software Technology 48 (2006) 1172–1186.
[22] A. Endo, A. Simao, A systematic review on formal testing approaches for web services, in: Brazilian Workshop on Systematic and Automated Software Testing, International Conference on Testing Software and Systems, 2010.
[23] E. Engstrom, P. Runeson, Software product line testing – a systematic mapping study, Journal of Information and Software Technology 53 (2011) 2–13.
[24] E. Engstrom, P. Runeson, M. Skoglund, A systematic review on regression test selection techniques, Journal of Information and Software Technology 52 (2010) 14–30.
[25] V.T.N.N. Frank Elberzhager, Jürgen Münch, A systematic mapping study on the combination of static and dynamic quality assurance techniques, Inform. Softw. Technol. 54 (2012) 1–15.
[26] V. Garousi, A. Mesbah, A.B.-C.S. Mirshokraie, A Systematic Mapping of Web Application Testing: Online Repository <http://www.softqual.ucalgary.ca/projects/Web_Application_Testing_Repository/>.
[27] V. Garousi, T. Varma, A bibliometric assessment of canadian software engineering scholars and institutions (1996–2006), Canadian Journal of Computer and Information Science 3 (2) (2010) 19–29.
[28] J.J. Garrett, Ajax: A New Approach to Web Applications, February 2005 <http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications> (visited 27.01.2012).

[29] R.L. Glass, T.Y. Chen, An assessment of systems and software engineering scholars and institutions (1998–2002), Journal of Systems and Software 68 (1) (2003) 77–84.
[30] R.L. Glass, T.Y. Chen, An assessment of systems and software engineering scholars and institutions (1999–2003), Journal of Systems and Software 76 (1) (2005) 91–97.
[31] M. Grindal, J. Offutt, S. F Andler, Combination testing strategies: a survey, Software Testing, Verification, and Reliability 15 (2005) 167–199.
[32] T.D. Hellmann, A. Hosseini-Khayatand, F. Maurer, Agile Interaction Design and Test-Driven Development of User Interfaces – A Literature Review, vol. 9, Springer, 2010.
[33] Y. Jia, M. Harman, An analysis and survey of the development of mutation testing, IEEE Transactions of Software Engineering 37 (5) (2011) 649–678.
[34] Y. Jia, M. Harman, Mutation Testing Repository <http://www.dcs.kcl.ac.uk/pg/jiayue/repository> (accessed April 2012).
[35] N. Juristo, A.M. Moreno, S. Vegas, Reviewing 25 years of testing technique experiments, Empirical Software Engineering 9 (2004) 7–44.
[36] B. Kitchenham, D. Budgen, P. Brereton, The value of mapping studies - a participant-observer case study, in: Proceedings of Evaluation and Assessment in Software Engineering, 2010.
[37] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.
[38] B. Kitchenham, T. Dyba, M. Jorgensen, Evidence-based software engineering, in: Proceedings of 26th International Conference on Software Engineering (ICSE), IEEE Computer Society, 2004, pp. 273–281.
[39] B.A. Kitchenham, D. Budgen, O.P. Brereton, Using mapping studies as the basis for further research: a participant-observer case study, Journal of Information and Software Technology 53 (2011) 638–651.
[40] B. Marin, T. Vos, G. Giachetti, A. Baars, P. Tonella, Towards testing future web applications, in: Proceedings 5th International Conference on Research Challenges in Information Science (RCIS), IEEE, 2011, pp. 1–12.
[41] P. McMinn, Search-based software test data generation: a survey, Software Testing, Verification and Reliability 14 (2) (2004).
[42] P. McMinn, Search-based software test data generation: a survey: research articles, Software Testing, Verification and Reliability 14 (2004) 105–156.
[43] A.M. Memon, B.N. Nguyen, Advances in automated model-based system testing of software applications with a GUI front-end, Advances in Computers, vol. 80, Academic Press, 2010.
[44] C.R.L. Neto, P.A. da Mota Silveira Neto, E.S. de Almeida, S.R. de Lemos Meira, A mapping study on software product lines testing tools, in: Proceedings of International Conference on Software Engineering and Knowledge Engineering, 2012.
[45] M. Palacios, J. García-Fanjul, J. Tuya, Testing in service oriented architectures with dynamic binding: a mapping study, Information Software and Technology 53 (2011) 171–189.
[46] C.S. Pasareanu, W. Visser, A survey of new trends in symbolic execution for software testing and analysis, International Journal on Software Tools for Technology Transformation 11 (4) (2009) 339–353.
[47] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2008, pp. 71–80.
[48] J. Ren, R.N. Taylor, Automatic and versatile publications ranking for research institutions and scholars, Communications of the ACM 50 (6) (2007) 81–85.
[49] A. van Deursen, A. Mesbah, Research issues in the automated testing of Ajax applications, in: Proceedings 36th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'10), Lecture Notes in Computer Science, vol. 5901, Springer-Verlag, 2010, pp. 16–28.
[50] J.A. Whittaker, Exploratory Software Testing, Pearson Education, 2009.
[51] Z. Zakaria, R. Atan, A. Ghani, N. Sani, Unit testing approaches for bpel: a systematic review, in: Proceedings of the Asia–Pacific Software Engineering Conference, 2009.
[52] Y. Zhang, Repository of Publications on Search based Software Engineering <http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/> (April 2012).
[53] Systematic Mapping References (79 papers) M. Alalfi, J. Cordy, T. Dean, Automating coverage metrics for dynamic web applications, in: Proceedings of the 14th European Conference on Software Maintenance and Reengineering (CSMR), IEEE Computer Society, 2010, pp. 51–60.
[54] N. Alshahwan, M. Harman, Automated session data repair for web application regression testing, in: Proceedings of the 1st International Conference on Software Testing, Verification, and Validation (ICST), IEE Computer Society, 2008, pp. 298–307.
[55] N. Alshahwan, M. Harman, R. Marchetto, P. Tonell, Improving web application testing using testability measures, in: Proceedings of 11th IEEE International Symposium on Web Systems Evolution (WSE), IEE Computer Society, 2009, pp. 49–58.
[56] D. Amalfitano, A.R. Fasolino, P. Tramontana, Rich internet application testing using execution trace data, in: Proceedings of the 3rd International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), IEE Computer Society, 2010, pp. 274–283.
[57] D. Amyot, J. Roy, M. Weiss, UCM-driven testing of web applications, in: Proceedings of the 12th International SDL Forum, Springer, 2005, pp. 247–264.
[58] A. Andrews, J. Offutt, R. Alexander, Testing web applications by modeling with FSMs, Software Systems and Modeling (SoSYM) 4 (3) (2005) 326–345.
[59] A. Andrews, J. Offutt, C. Dyreson, C. Mallery, K. Jerath, R. Alexander, Scalability issues with using FSMWeb to test web applications, Information and Software Technology (IST) 52 (1) (2010) 52–66.
[60] S. Artzi, J. Dolby, S.H. Jensen, A. Møller, F. Tip, A framework for automated testing of javascript web applications, in: Proceeding of the 33rd International Conference on Software Engineering, ICSE '11, ACM, 2011, pp. 571–580.
[61] S. Artzi, J. Dolby, F. Tip, M. Pistoia, Practical fault localization for dynamic web applications, in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE), ACM, 2010, pp. 265–274.
[62] S. Artzi, A. Kiezun, J. Dolby, F. Tip, D. Dig, A. Paradkar, Finding bugs in dynamic web applications, in: Proceedings of the International Symposium on Software Testing and Analysis (ISSTA), ACM, 2008, pp. 261–272.
[63] H. Bajwa, W. Xiong, F. Maurer, Evaluating current testing processes of web-portal applications, in: Proceedings of the International Conference Web Engineering (ICWE), Springer, 2005, pp. 603–605.
[64] C. Bellettini, A. Marchetto, A. Trentini, TestUml: User-metrics driven web applications testing, in: Proceedings of the 2005 ACM Symposium on Applied Computing (SAC), ACM, 2005, pp. 1694–1698.
[65] M. Benedikt, J. Freire, P. Godefroid, VeriWeb: Automatically testing dynamic web sites, in: Proceedings of the 11th International World Wide Web Conference (WWW), ACM, 2002.
[66] B. Bordbar, K. Anastasakis, MDA and analysis of web applications, in: Proceedings of the 2nd International Conference on Trends in Enterprise Application Architecture (TEAA), Springer, 2006, pp. 44–45.
[67] E.C.B. de Matos, T. Sousa, From formal requirements to automated web testing and prototyping, Innovations in Systems and Software Engineering (ISSE) 6 (1–2) (2010) 163–169.
[68] K. Dobolyi, E. Soechting, W. Weimer, Automating regression testing using web-based application similarities, International Journal on Software Tools for Technology Transfer (STTT) 13 (9) (2011) 111–129.
[69] K. Dobolyi, W. Weimer, Modeling consumer-perceived web application fault severities for testing, in: Proceedings of the 19th International Symposium on Software Testing and Analysis (ISSTA), ACM, 2010, pp. 97–106.
[70] S. Elbaum, G. Rothermel, S. Karre, M. Fisher II, Leveraging user-session data to support web application testing, IEEE Transactions on Software Engineering 31 (3) (2005) 187–202.
[71] J. Ernits, R. Roo, J. Jacky, M. Veanes, Model-based testing of web applications using NModel, in: Proceedings of IFIP International Conference on Testing of Software and Communication Systems and International FATES Workshop (TESTCOM-FATES), Springer, 2009, pp. 211–216.
[72] T. Ettema, C. Bunch, 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 (ASE), ACM, 2010, pp. 235–244.
[73] Y. Gerlits, Testing ajax functionality with UniTESK, in: Proceedings of the 4th Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE), 2010, pp. 50–57.
[74] H.B.K.T.H. Liu, Testing input validation in web applications through automated model recovery, Journal of Systems and Software (JSS) 81 (2) (2008) 222–233.
[75] h. Raffelt, B. Steffen, T. Margaria, M. Merten, Hybrid test of web applications with webtest, in: Proceedings of the Workshop on Testing, Analysis, and Verification of Web Services and Applications (TAV-WEB), ACM, 2008, pp. 1–7.
[76] W. Halfond, S. Anand, A. Orso, Precise interface identification to improve testing and analysis of web applications, in: Proceedings of the Eighteenth International Symposium on Software Testing and Analysis (ISSTA'09), ACM, 2009, pp. 285–296.
[77] W. Halfond, A. Orso, Improving test case generation for web applications using automated interface discovery, in: Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC-FSE), ACM, 2007, pp. 145–154.
[78] W. Halfond, A. Orso, Automated identification of parameter mismatches in web applications, in: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), ACM, 2008, pp. 181–191.
[79] J. Hao, E. Mendes, Usage-based statistical testing of web applications, in: Proceedings of the 6th International Conference on Web Engineering (ICWE), ACM, 2006, pp. 17–24.
[80] S.H. Jensen, M. Madsen, A. Moller, Modeling the HTML DOM and browser API in static analysis of JavaScript web applications, in: Proceedings of ACM SIGSOFT Symposium and European conference on Foundations of Software Engineering (ESEC-FSE), ACM, 2011, pp. 59–69.
[81] C. Kallepalli, J. Tian, Measuring and modeling usage and reliability for statistical web testing, IEEE Transactions on Software Engineering (TSE) 27 (11) (2001) 1023–1036.
[82] P. Koopman, P. Achten, R. Plasmeijer, Model-based testing of thin-client web applications and navigation input, in: Proceedings of the 10th International Conference on Practical Aspects (PADL), Springer, 2008, pp. 299–315.
[83] N. Li, T. Xie, M. Jin, C. Liu, Perturbation-based user-input-validation testing of web applications, Journal of Systems and Software (JSS) 83 (11) (2010) 2263–2274.

[84] D. Licata, S. Krishnamurthi, Verifying interactive web programs, in: Proceedings of the 19th International Conference on Automated Software Engineering (ASE), IEEE Computer Society, 2004, pp. 164–173.
[85] C. Liu, Data flow analysis and testing of JSP-based web applications, Information and Software Technology (IST) 48 (12) (2006) 1137–1147.
[86] G.D. Lucca, A. Fasolino, F. Faralli, Testing web applications, in: Proceedings of the International Conference on Software Maintenance (ICSM), IEEE Computer Society, 2002, pp. 310–319.
[87] G.D. Lucca, A. Fasolino, P. Tramontana, A technique for reducing user session data sets in web application testing, in: Proceedings of the 8th IEEE International Symposium on Web Site Evolution (WSE), IEEE Computer Society, 2006, pp. 7–13.
[88] X. Luo, F. Ping, M. Chen, Clustering and tailoring user session data for testing web applications, in: Proceedings of the International Conference on Software Testing Verification and Validation (ICST), IEEE Computer Society, 2009, pp. 336–345.
[89] N. Mansour, M. Houri, Testing web applications, Information and Software Technology (IST) 48 (1) (2006) 31–42.
[90] A. Marchetto, Talking about a mutation-based reverse engineering for web testing: a preliminary experiment, in: Proceedings of the 2008 Sixth International Conference on Software Engineering Research, Management and Applications (SERA), IEEE Computer Society, 2008, pp. 161–168.
[91] A. Marchetto, F. Ricca, P. Tonella, Empirical validation of a web fault taxonomy and its usage for fault seeding, in: Proceedings of the 9th IEEE International Workshop on Web Site Evolution (WSE), IEEE Computer Society, 2007, pp. 31–38.
[92] A. Marchetto, F. Ricca, P. Tonella, A case study-based comparison of web testing techniques applied to AJAX web applications, International Journal on Software Tools for Technology Transfer (STTT) 10 (6) (2008) 477–492.
[93] A. Marchetto, R. Tiella, P. Tonella, N. Alshahwan, M. Harman, Crawlability metrics for automated web testing, International Journal on Software Tools for Technology Transfer (STTT) 13 (2) (2011) 131–149.
[94] A. Marchetto, P. Tonella, Using search-based algorithms for ajax event sequence generation during testing, Empirical Software Engineering (ESE) 16 (1) (2011) 103–140.
[95] A. Marchetto, P. Tonella, F. Ricca, State-based testing of ajax web applications, in: Proceedings of the 1st International Conference on Software Testing, Verification, and Validation (ICST), IEEE Computer Society, 2008, pp. 121–130.
[96] A. Mesbah, M.R. Prasad, Automated cross-browser compatibility testing, in: Proceeding of the 33rd International Conference on Software Engineering, ICSE '11, ACM, 2011, pp. 561–570.
[97] A. Mesbah, A. van Deursen, Invariant-based automatic testing of Ajax user interfaces, in: Proceedings of the 31st International Conference on Software Engineering (ICSE'09), IEEE Computer Society, 2009, pp. 210–220.
[98] Y. Minamide, Static approximation of dynamically generated web pages, in: Proceedings of the 14th International Conference on World Wide Web (WWW), ACM, 2005, pp. 432–441.
[99] J. Offutt, Y. Wu, Modeling presentation layers of web applications for testing, Software Systems Modeling (SoSYM) 9 (2) (2010) 257–280.
[100] J. Offutt, Y. Wu, X. Du, H. Huang, Bypass testing of web applications, in: Proceedings of the 15th International Symposium on Software Reliability Engineering (ISSRE), IEEE Computer Society, 2004, pp. 187–197.
[101] M. Ozkinaci and A. betin Can, Detecting execution and html errrors in ASP.Net web applications, in: Proceedings of the 6th International Conference on Software and Data Technologies (ICSOFT), 2011, pp. 172–178.
[102] K. Pattabiraman, B. Zorn, DoDOM: Leveraging DOM invariants for Web 2.0 application robustness testing, in: Proceedings of thel International Symposium on Software Reliability Engineering (ISSRE), IEEE Computer Society, 2010.
[103] U. Praphamontripong and J. Offutt, Applying mutation testing to web applications, in: Proceedings of the 3rd International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), 2010, pp. 132–141.
[104] Y. Qi, D. Kung, E. Wong, An agent-based data-flow testing approach for web applications, Information and Software Technology (IST) 48 (12) (2006) 1159–1171.
[105] L. Ran, C. Dyreson, A. Andrews, R. Bryce, C. Mallery, Building test cases and oracles to automate the testing of web database applications, Information and Software Technology (IST) 51 (2) (2009) 460–477.
[106] F. Ricca, P. Tonella, Analysis and testing of web applications, in: Proceedings of the 23rd International Conference on Software Engineering (ICSE), IEEE Computer Society, 2001, pp. 25–34.
[107] F. Ricca, P. Tonella, Construction of the system dependence graph for web application slicing, in: Proceedings of the 2nd IEEE International Workshop on Source Code Analysis and Manipulation (SCAM), IEEE Computer Society, 2002, pp. 123–132.
[108] F. Ricca, P. Tonella, Testing processes of web applications, Annals of Software Engineering (ASE) 14 (1-4) (2002) 93–114.
[109] D. Roest, A. Mesbah, A. van Deursen, Regression testing ajax applications: coping with dynamism, in: Proceedings of the 3rd International Conference on Software Testing, Verification and Validation (ICST'10), IEEE Computer Society, 2010, pp. 128–136.
[110] S. Sampath, R. Bryce, G. Viswanath, V. Kandimalla, A.G. Koru, Prioritizing user-session-based test cases for web application testing, in: Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation (ICST), IEEE Computer Society, 2008, pp. 141–150.
[111] S. Sampath, V. Mihaylov, A. Souter, L. Pollock, A scalable approach to user-session based testing of web applications through concept analysis, in: Proceedings of the 19th IEEE International Conference on Automated Software Engineering (ASE), IEEE Computer Society, 2004, pp. 132–141.
[112] S. Sampath, S. Sprenkle, E. Gibson, L. Pollock, Integrating customized test requirements with traditional requirements in web application testing, in: Proceedings of the Workshop on Testing, Analyis and Verification of Web Services and Applications (TAV-WEB), ACM, 2006, pp. 23–32.
[113] S. Sampath, S. Sprenkle, E. Gibson, L. Pollock, Web application testing with customized test requirements – an experimental comparison study, in: Proceedings of the 17th International Symposium on Software Reliability Engineering (ISSRE), IEEE Computer Society, 2006, pp. 266–278.
[114] S. Sampath, S. Sprenkle, E. Gibson, L. Pollock, Applying concept analysis to user-session-based testing of web applications, IEEE Transactions on Software Engineering (TSE) 33 (10) (2007) 643–658.
[115] S. Sampath, V. Mihaylov, A. Souter, L. Pollock, Composing a framework to automate testing of operational web-based software, in: Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM), IEEE Computer Society, 2004, pp. 104–113.
[116] P. Saxena, D. Akhawe, S. Hanna, F. Mao, S. McCamant, D. Song, A symbolic execution framework for JavaScript, in: Proceedings of the 31st IEEE Symposium on Security and Privacy (SP), IEEE Computer Society, 2010, pp. 513–528.
[117] E.D. Sciascio, F. Donini, M. Mongiello, R. Totaro, D. Castelluccia, Design verification of web applications using symbolic model checking, in: Proceedings of the 5th International Conference Web Engineering (ICWE), Springer, 2005, pp. 69–74.
[118] S. Sprenkle, H. Esquivel, B. Hazelwood, L. Pollock, WebVizOr: a visualization tool for applying automated oracles and analyzing test results of web applications, in: Proceedings of the Testing: Academic and Industrial Conference - Practice and Research Techniques (TAIC-PART), IEEE Computer Society, 2008, pp. 89–93.
[119] S. Sprenkle, L. Pollock, H. Esquivel, B. Hazelwood, S. Ecott, Automated oracle comparators for testing web applications, in: Proceedings of the 18th IEEE International Symposium on (ISSRE), IEEE Computer Society, 2007, pp. 117–126.
[120] S. Sprenkle, L. Pollocky, L. Simko, A study of usage-based navigation models and generated abstract test cases for web applications, in: Proceedings of the 4th IEEE International Conference on Software Testing, Verification and Validation (ICST), IEEE Computer Society, 2011, pp. 230–239.
[121] S. Sprenkle, S. Sampath, E. Gibson, L. Pollock, A. Souter, An empirical comparison of test suite reduction techniques for user-session-based testing of web applications, in: Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM), IEEE Computer Society, 2005, pp. 587–596.
[122] B. Stepien, L. Peyton, P. Xiong, Framework testing of web applications using TTCN-3, International Journal on Software Tools for Technology Transfer (STTT) 10 (4) (2008) 371–381.
[123] A. Tappenden, J. Miller, Cookies: A deployment study and the testing implications, ACM Transactions on the Web (TWEB) 3 (3) (2009) 1–49.
[124] J. Tian, L. Ma, Web testing for reliability improvement, Advances in Computers 67 (2006) 178–225.
[125] P. Tonella, F. Ricca, A 2-layer model for the white-box testing of web applications, in: Proceedings of the 6th IEEE International Workshop on Web Site Evolution (WSE), IEEE Computer Society, 2004, pp. 11–19.
[126] P. Tonella, F. Ricca, Statistical testing of web applications, Journal of Software Maintenance and Evolution: Research and Practice (JSME) 16 (1-2) (2004) 103–127.
[127] P. Tonella, F. Ricca, Web application slicing in presence of dynamic code generation, Automated Software Engineering (ASE) 12 (2) (2005) 259–288.
[128] G. Wassermann, D. Yu, A. Chander, D. Dhurjati, H. Inamura, Z. Su, Dynamic test input generation for web applications, in: Proceedings of the International symposium on Software testing and analysis (ISSTA), ACM, 2008, pp. 249–260.
[129] W. Xiong, H. Bajwa, F. Maurer, WIT: A framework for in-container testing of web-portal applications, in: Proceedings of the 5th International Conference on Web Engineering (ICWE), Springer, 2005, pp. 87–97.
[130] J. Yang, J. Huang, F. Wang, W. Chu, Constructing an object-oriented architecture for web application testing, Journal of Information Science and Engineering (JISE) 18 (1) (2002) 59–84.
[131] Y. Zheng, T. Bao, X. Zhang, Statically locating web application bugs caused by asynchronous calls, in: Proceedings of the 20th International Conference on World Wide Web (WWW), ACM, 2011, pp. 805–814.