The Journal of Systems and Software 219 (2025) 112186

Contents lists available at ScienceDirect

The Journal of Systems & Software

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

Web application testing—Challenges and opportunities☆

Sebastian Balsam *, Deepti Mishra

Department of Computer Science, Norwegian University of Science and Technology, Norway

ARTICLE INFO

Keywords:
Systematic literature review
Web application
Testing
Test models
Testing tools

ABSTRACT

Context: A large part of the software produced by many companies and organizations today are web applications. Testing web applications is vital to ensure and maintain the quality of these systems. They play an important role in promoting brands and enabling better communication with customers.
Objective: There is a gap in existing literature research in recent years regarding testing of web applications, although the landscape of web applications techniques has changed. New methods, frameworks, environments and techniques have recently been used both for developing and testing these applications. This paper presents an overview of the research directions, problems and challenges in the field of Web application testing in the last decade. Our paper investigates current implementation and validation techniques, the quality of existing approaches and reveals areas of incomplete or superficial research.
Methods: In this paper, a systematic literature review about the state of web application testing has been conducted. Based on initially about 6000 papers, that were extracted from Science Direct, Springer Link, WebOfScience, IEEE Explore and ACM, we used a final number of 72 papers after a filtration process for this literature review. The extracted papers were examined for demographics, problems, techniques and tools. We looked at the quality, the empirical evidence and the test application used for validating the different methods in the extracted papers.
Results: The most important journals, authors, tools and research directions in this field are discovered, a deep analysis of quality, rigor and empirical evidence is given, and the most important validation applications are described. We found that three groups of authors contributed to more than 25% of the papers, the three most important journals published more than 50% of the papers. 30% of the developed tools were open accessible. Most papers had a good description of study design and threats for validity, but have little industrial relevance. Only 6 papers validated on industrial applications. Only 40% of these papers compared their technique with other existing techniques, and applications used for validation are usually outdated.
Conclusions: We discuss trends and challenges in research in web application testing. We also show gaps in research and areas that need more attention from the research community. Research in web application testing needs more focus on industrial relevance and scalability to analyze the usability for industry. New techniques should be validated on modern test application frameworks to get comparable results. The results can help researchers to get an overview of publication venues, active researchers, current research gaps and problems in the field.

1. Introduction

Web applications have become an integral part of many companies and organizations. By using web applications, user engagement can be increased as companies can get in contact with them through simple and easily accessible means (Wirtz et al., 2010). Furthermore, data can be stored centrally, making it much easier to use for different purposes. By storing the data in the cloud, it also increases security, makes customization and scalability easier, and simplifies installation and maintenance. Often, web applications are the main form of interaction for users to control their data and flow of data through the system to get the desired result.

It is therefore necessary for developers to test these applications extensively and thoroughly to maintain quality. However, this leads to various challenges. The user interacts mostly through web pages, which present a very rich but rather complex user interface. Often, a user has different choices for interaction at the same time, which results in multiple testing paths. Furthermore, webpages are rendered on different devices in different sizes and designs, each with their

☆ Editor: Professor Laurence Duchien.

* Corresponding author.
E-mail addresses: sebastian.balsam@posteo.de (S. Balsam), deepti.mishra@ntnu.no (D. Mishra).

https://doi.org/10.1016/j.jss.2024.112186
Received 21 February 2024; Received in revised form 9 June 2024; Accepted 14 August 2024
Available online 17 August 2024

own challenges. Often, the same information is presented in different views and forms (Jameel et al., 2018; Hoffswell et al., 2020). Another problem is that server code is usually not written in the same programming language as the browser code, what makes it difficult to test both of them as a whole. Aside from that, developers have to deal with communication problems between the browser and the web server, since information has to flow through the internet, with all its drawbacks (Madhurima et al., 2015). Testing web applications is therefore usually more complex than testing other software products, and methods and tools must be found to make testing more efficient.

Even though research seems to be distant from the work of practitioners in an industrial context (Garousi and Felderer, 2017), much work has been done from the academic side to help web application testing in the industry.

Doğan et al. (2014) did a systematic literature review (SLR) that delivered an overview of the state of the art of web application testing. Their work was based on another systematic mapping study (Garousi et al., 2013) and presents insight into used testing tools, testing techniques, and observed bug taxonomies.

Since their study in 2014, to the best of the authors' knowledge, there have been no general systematic literature reviews on web application testing. Therefore, we think there is a need for a follow-up investigation on this subject. Firstly, both web development techniques and web testing techniques have changed substantially in the last decade. Even though there existed web frameworks before 2014, the use of such frameworks like Angular JS or React has increased. Ajax has become a standard in web applications since its first draft in 2005 (Garrett, 2005). The architecture of web applications has changed as well. It is more common to have single-page applications, and there are new possibilities with browser-based web storage for such applications to work even offline. Cross-platform applications (El-Kassas et al., 2017) and progressive web apps are another relative new development in the field of web applications (Biørn-Hansen et al., 2017).

We are carrying out a new systematic literature review on 72 papers between 2013 and 2023 with research questions, mainly based on the work by Doğan et al. (2014), but with additional and revised research questions to adjust to the changes in the field of web application testing since 2013. The research questions we want to look at are:

- Which authors and journals are most important?
- Which problems are addressed, and which tools and techniques have been used and developed?
- What is the quality of the studies?
- Which empirical evidence strengthens the studies?
- Which applications or datasets have been used for validation?

This paper is organized as follows: the next section will give an overview of the background and related work. In the method section, we will explain the extraction process and the development and implementation of our research. The last section presents the results that will answer our research questions in detail and finally a short summary is provided in the conclusion.

2. Background and related work

Kitchenham and Charters (2007) introduced systematic literature reviews to the field of software engineering. The benefits of a systematic literature review are to search thoroughly through all available publications with well-defined search objects and criteria for inclusion and exclusion and a transparent quality evaluation process. This process should be reproducible and should identify all relevant research areas, gaps and methods in a standardized way. In the context of web applications, we find a few literature reviews and some of them being systematic.

The SLR this review is mostly based on is by Doğan et al. (2014). They extracted 95 papers between 2000 and 2012 and did a review on different test tools, test methods, fault models, assessment metrics for quality, cost and relevance and empirical evidence and industrial relevance. Their review is a complimentary study to an earlier mapping study (Garousi et al., 2013). The only issue with this study is its relative age. It is the most comprehensive review we could find, and we will use it as a guidance for our review.

Besides that, there are some systematic reviews that are close to our field. Prokhorenko et al. (2016) and later Aydos et al. (2022) give an overview of the state of the art of security related testing methods. We will exclude security related testing papers in our review, even though they are related to pure error testing, since security testing usually has different approaches and goals. Dadkhah (2020) did a systematic literature review on semantic web enabled software testing. They included 52 primary studies about software testing that were based on the application of semantic web technologies. The scope of our review is different since we focus only on web applications.

Dalisay (Dalisay, 2018) gives a summary about the problems of mobile web apps in an SLR. It becomes clear, that at the time of writing, web apps still have some issues in contrast to native apps. The issues are described and evaluated in detail. Also connected to app testing, Kong et al. (2019) examined 102 papers in an SLR about testing android apps. Both reviews are peripheral relevant, since they clarify challenges in an adjacent area of web application testing.

Molina-Ríos (Molina-Ríos and Pedreira-Souto, 2020) describes development methods of web applications in their SLR. Even though this review gives a good overview over different development methodologies, testing is not a part of their review. Palomino et al. (2021) gives a literature review about web analytics for user experience. This review is mostly about the design and usability of web applications, not so much about errors and testing.

There are other papers that are not SLR's but still relevant as review studies. Li et al. (2014b) give a detailed overview of different testing techniques used in 25 analyzed papers. The identified techniques are model and graph based testing, mutation testing, search-based testing, scanning and crawling techniques, random testing, fuzz testing, concolic testing and user session based testing. The paper not only describes the techniques in detail, but also give an overview of the used evaluation methods like coverage and cost-effectiveness. As a survey paper of the different techniques it is very useful as a method for classification of techniques, but since it is from 2014, it should be investigated, which of these techniques are used in recent years. Alenzi et al. (2022) give an overview of the most important web testing frameworks. They identify seven of them (Selenium, Watir, TestComplete, SahiPro, QTP, JMeter and LoadRunner) and compare them in a short and concise way. The paper provides an idea of modern testing frameworks, even though it lacks many of them like those used in JavaScript, but at least is a good starting point for the research question.

As this review is focused on research papers, there is little discussion on the impact for the industry. Garousi and Felderer (2017) found gaps in what the industry wants, and what researchers focus on. Research that focuses on the actual needs from industry like in another paper by Garousi et al. (2020) or from Alshahwan et al. (2023) is needed.

The mentioned papers are either in a neighboring area, without exactly addressing the problems that are addressed here, or they are older and do not capture the studies in recent times. In addition, new challenges of the last decade pose new questions to the existing literature, to which we want to respond. We want to find out, in which direction research has developed in recent years, what gaps exist in research, and which research results have made significant contributions.

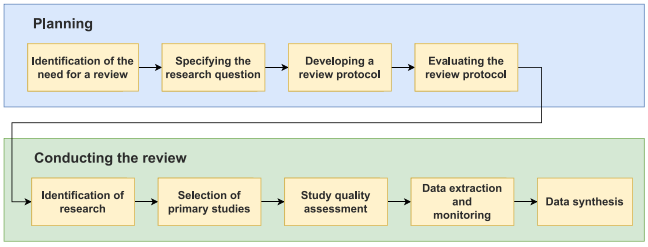


Fig. 1. Review process based on Kitchenham's systematic literature review process.

3. Research method

Kitchenham and Charters (2007) describe a process to perform a systematic literature review (SLR). We followed this guideline to perform our SLR as (Doğan et al., 2014) did in their paper. By doing this, we extended their review with new findings after 2013 and are able to compare the results in the conclusions. Our study setup differs in some points. We had some additional research questions and changed some methods of data extraction to adjust to the changes both in web development, but also to changes in the research journal landscape with different offers for data extraction. Our search protocol has been adjusted accordingly.

The review process we did is divided in the stages shown in Fig. 1. It is based on the methods by Doğan et al. (2014) and Brereton et al. (2007) and aims to provide an evidence based research review instead of an opinion based review (Kitchenham et al., 2004). The advantage with this approach is reproducibility and an objective view of the examined topic. The data extraction is carried out in a planned and organized form, and the extracted papers are classified and evaluated through a predefined set of rules. We followed this process and will present at first the setup for this review.

In the next sections, we will first formulate the research questions. They will be the guidelines, the extraction process has to be oriented to. We will then discuss the main points of the review protocol containing the search term, the resources to be searched along with exclusion and inclusion criteria, data extraction strategy and synthesis strategy. The data extraction will again focus on the research questions to gather qualitative and quantitative data.

3.1. Research questions

As previously mentioned, it has been ten years since the last SLR by Doğan et al. (2014) which will be used as a base for this review. But we will also introduce some changes. In recent years, the landscape of web applications have changed in some ways that will be mirrored in the results. RQ 2, RQ 3 and RQ 4 have been taken directly from Doğan et al. (2014). In addition, we also have some additional questions (RQ 1 and RQ 5), that were not a part of the original paper from 2014.

RQ 1: Demographic questions: How much research has been conducted in recent years and by which authors and journals? By answering this question, it becomes easier for a researcher to start exploring the field. It becomes clearer which authors and journals are focusing on this research area.

- RQ 1.1 : How many papers have been published per year?
- RQ 1.2 : Who are the main contributors in the field? Which countries are they from?
- RQ 1.3 : Which journals are the most important in the field?

RQ 2: Problems, Testing techniques & Tools: What type of testing techniques have been researched and which tools have been proposed? We want to know in which direction research has been moving. What are the main challenges and possible solutions?

- RQ 2.1 : Which problems in testing are being addressed?
- RQ 2.2 : What types of input/inferred test models have been proposed/used?
- RQ 2.3 : What types of fault models related to web applications have been used?
- RQ 2.4 : Which tools have been proposed, and what are their capabilities?

RQ 3: Design and report of the empirical studies: What is the quality of the studies? Are they properly designed and easy to replicate? This question can reveal problems with the proposed methods that need further research.

- RQ 3.1 : Which metrics are used for assessing cost and effectiveness of web application testing techniques?
- RQ 3.2 : What are the threats to validity in the empirical studies?
- RQ 3.3 : What is the level of rigor and industrial relevance of the empirical studies?

RQ 4: Empirical evidence: What is the empirical evidence of scalability of the proposed techniques. This question, together with RQ 3 will show the usability of the techniques beyond research.

- RQ 4.1 : Is there any evidence regarding the scalability of the web application techniques?
- RQ 4.2 : Have different techniques been empirically compared with each other?
- RQ 4.3 : How much empirical evidence exists for each category of techniques and type of web apps?

RQ 5: Data: What kind of test applications are most frequently used for web interface testing? Are there any applications that are state of the art for validation of the proposed methods?

- RQ 5.1 : Which test applications are used for validation? Which languages is used for implementation?
- RQ 5.2 : Are these datasets open accessible?

3.2. Search term and resources to be searched

For the search term, we wanted to get papers that include both results for testing or analyzing, and results for web applications. Since we already found some papers with the term "user interface", we added the term to the search string together with "web", "website" and "web application". The general search string therefore is:

```
( web OR website OR "web application"
 OR "user interface" )
AND
( test OR testing OR analysis OR analyzing
 OR verification)
```

In addition, we only included papers between 2013 and 2023 since the paper by Doğan et al. (2014) already examines papers until 2013. We searched on title, abstract and keywords with this search string. Our SLR uses the resources shown in Table 1 to find papers.

ACM Digital Library allowed to search for the search string, with filters for publication year and publication type. The results could be

Table 1
Resources to search for papers for extraction.

Resource	URL	Number of papers
ACM Digital Library	https://portal.acm.org	529
IEEE Explore	https://ieeexplore.ieee.org	2000
ScienceDirect	https://www.sciencedirect.com	899
Springer Link	https://link.springer.com/	1000
WebOfScience	https://www.webofscience.com/	1000

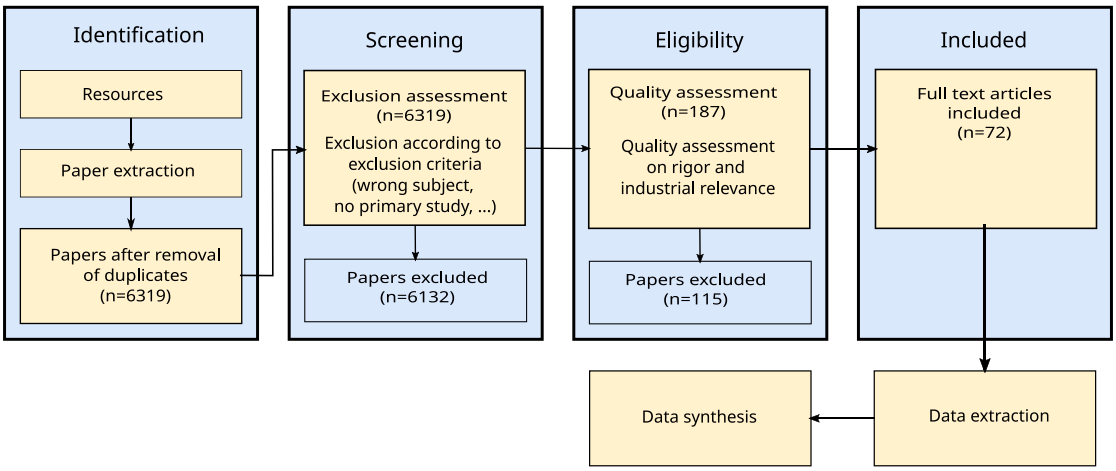


Fig. 2. Extraction process in form derived from PRISMA.

downloaded as Bibtex citations that we could further process to extract the relevant data. *IEEE Explore* and *WebOfScience* allowed about the same search criteria and a download option as a csv file that could be further processed. *Science Direct* and *Springer Link* allowed only shorter search strings, we therefore reduced the search string by stripping the logical operators resulting in logical OR's between words. We could filter the results for publication date and publication type and export the results from Science Direct again as Bibtex citations for further processing. The results from Springer Link could be exported directly as a csv file.

After we had extracted all papers, we converted the Bibtex files to a general csv format to make all extracted papers comparable, this includes *i.e.* conversions of commas or semicolon to separate authors. In this way we got 5 separate lists of publications from the extraction resources in a comparable format.

3.3. Exclusion criteria and quality assessment

To filter the extracted papers further, usually inclusion or exclusion criteria are used to guarantee a certain quality even before reading the papers. These criteria can be applied to some extent automatically by filtering the results in a script. The following exclusion criteria are used in our study to filter out papers that are not relevant for our review:

- EC1 : paper is written in a language other than English.
- EC2 : full text not accessible.
- EC3 : paper is a duplicate or similar version.
- EC4 : paper is not a primary study.
- EC5 : paper is older than 2013.
- EC6 : paper is not peer-reviewed.
- EC7 : paper is not about web application testing techniques, we excluded also papers about utilization, security, performance and API testing.

In total, we extracted 5158 journal papers from all five sources and additional 1161 conference papers from ACM Digital Library and IEEE Explore. They were downloaded either in csv format or as a list of bibtex entries. We used python scripts to filter out duplicates and compile these lists into a common format to simplify the later merging process. Each of our chosen sources delivered a list of possible papers ordered by relevance for our search criteria according to the internal search algorithms for each source. That means, the most relevant papers were ordered first.

We searched through each list per resource starting from the most relevant papers, when we found 100 consecutive papers not relevant in respect to our search criteria, we stopped the filter process. This means we have not searched through all 6319 papers, but stopped when the results were not relevant anymore. The filtered lists per resource were then combined into one list of relevant papers. In this way, we could reduce the number of papers to 187.

Regarding the quality assessment, we applied a scoring scheme proposed by Ivarsson and Gorschek (2011) that assessed rigor and industrial relevance for each study. For evaluating the rigor of a study, we gave 0, 0.5 or 1 point for the description of context, description of study design and the discussion of validity as shown in Table 7. To evaluate industrial relevance, we gave either 1 or 0 points for the representativity of the evaluation subjects, evaluation context, the scale of the applications used for evaluation and the research methods. Only studies with more than 0 points in either or both dimensions of rigor and industrial relevance have been examined further. The screening process was carried out by one of the authors. The other author checked the assessment and in case of disagreement, an assessment was negotiated.

After assessment of quality, further papers had been filtered out and the number was reduced to 72 papers for further processing. This extraction process is shown in Fig. 2 in the form of a flow diagram influenced by PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) as it is usual for systematic literature reviews.

3.4. Data extraction and synthesis strategy

To address the research questions, we had to extract information in a measurable form. This means we can either extract information in a boolean or numerable value or in the form of a classification assignment. For each investigated paper, we extracted values described in Table 2 to make it possible to compare and classify the papers regarding the different research questions. For string values, we extracted values most closely found in the original text or gave multiple labels, to reflect the directions of the paper for that category. In the synthesis process, we then generalized the values for classification and comparison as described in Cruzes and Dyba (2011).

As synthesis strategy, we will give both quantitative results and qualitative answers to the research questions by using the classification results.

4. Results and discussions

In this section, we will gather the results to our research questions from the data extracted from the chosen papers based on the evaluation metrics and classifications described in Section 3.4. We will go through each research question and give both quantitative and qualitative results.

Table 2
A description of classification and evaluation metrics to address the research question.

Nr	Data item	RQ	Type	Description/Example
1	year of publication	1.1	Integer	year of publication
2	authors	1.2	String	list of authors
3	journal	1.3	String	journal name
4	types of test models	2.1	String	exploration testing, model based testing, ...
5	types of fault models/bug taxonomy	2.2	String	general, X-path changes, Javascript faults...
6	tools have been proposed	2.3	String	Similio, Tapir, Robula+, ...
7	Accessibility of tools	2.3	String	URL
8	metrics: Cost metrics	3.1	String	time efficiency, model size
9	metrics: Effectiveness metrics:	3.1	String	code coverage, faults detected
10	metrics: Other metrics:	3.1	String	precision/ recall, Number of DOM violations
11	# threats to validity	3.2	Integer	How many threats are mentioned?
12	threats to validity	3.2	String	list of threats
13	level of rigor	3.3	Integer	scoring like in Ivarsson and Gorschek (2011)
14	level of industrial relevance	3.3	Integer	scoring like in Ivarsson and Gorschek (2011)
15	evidence regarding the scalability	4.1	Boolean	was this a topic in the study?
16	comparison of techniques	4.2	Boolean	Did the study compare techniques?
17	theme of technique	4.2	String	Javascript, Model based, others.
18	location for test evaluation	4.3	String	server or client side
19	static vs dynamic evaluation	4.3	String	static, dynamic or both
20	Ajax or synchronous	4.3	String	ajax, synchronous, both
21	validation test applications	5.1	String	list of test applications for validation
22	accessibility of test application	5.1	Boolean	are they accessible?

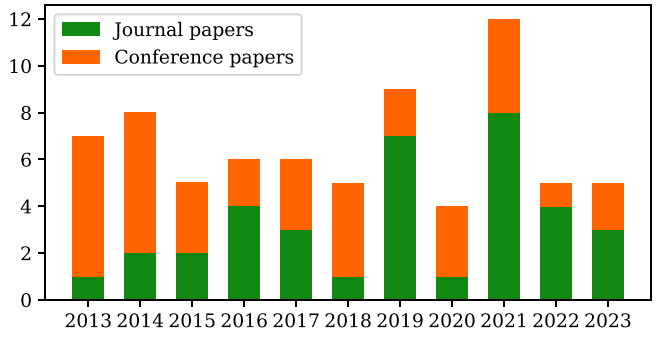


Fig. 3. Papers published per year. Since the year 2023 is not completed at the time of writing, this number is not comparable with the rest of the data.

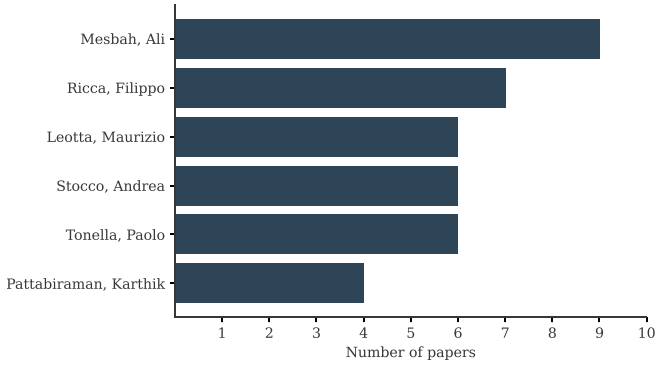


Fig. 4. Authors with more than 3 published papers in our selection of extracted papers.

4.1. RQ 1 — Demographic Questions

In this subsection, we want to examine the number of papers, the authors and the journals that published the extracted papers. By analyzing these, we hope to give directions and tendencies on which journals and authors are most interesting in our field.

4.1.1. RQ 1.1 : How many papers have been published per year?

The number of papers considered is too small to make statistically significant statements and trends about the development of the number of papers on the topic of web application testing as shown in Fig. 3.

> It is not conclusive if the numbers of papers per year about web application testing are increasing or declining.

4.1.2. RQ 1.2 : Who are the main contributors in the field?

Fig. 4 shows the authors with the most published papers in our selection of extracted papers. We can identify some key actors in our studied field. The author with the most papers is Ali Mesbah from Canada. He worked as coauthor on 9 papers (12%) together with, among others, K. Pattabiraman and F.S. Ocariza and A.M. Fard

Then, there is a group led by Paolo Tonella, including F. Ricca, M. Leotta, and A. Stocco from Italy. 7 papers come from this group alone (Leotta et al., 2016, 2021, 2018, 2023; Stocco et al., 2017; Biagiola et al., 2019a,b). Their research areas are stable X-path locators for regression testing, mutation testing as a tool to evaluate effectiveness of existing test suites and page object generation to make it easier to develop new tests.

Another group is formed around S. Sherin, M.Z. Iqbal, and M.U. Khan from Pakistan (Sherin et al., 2021, 2023). They worked on exploration strategies and coverage criteria and added further 2 papers. Khan and Iqbal also participated in a third paper (Imtiaz et al., 2021). They worked on coverage criteria used for web application testing and developed QExplore, an exploration strategy for web applications. Mesbah and his coauthors and both of groups from Italy and Pakistan together have authored 18 of our 71 papers and therefore more than 25 percent of papers in our extracted subset. This shows that they are important groups for web application testing.

Fig. 5 shows the frequencies of countries for the extracted papers. Most of the papers come from authors in the US, followed by Italy, China and Canada.

> We could identify few research groups that contribute in a significant way for web application testing. Three of the groups that contributed with most of the papers, wrote more than 25 percent of all extracted papers.

4.1.3. RQ 1.3 : Which journals are most important in the field?

Fig. 6 shows an overview of the most important journals and number of papers for each journal. The top 3 journals have published almost 50 percent of all papers included in this review. This shows how important they are for the field of software application testing. The

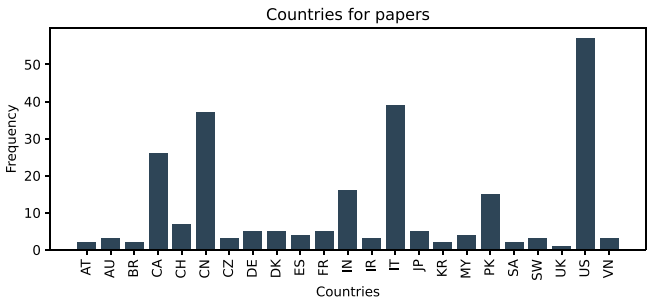


Fig. 5. Frequencies of countries for the extracted papers. We counted the countries of each author per paper.

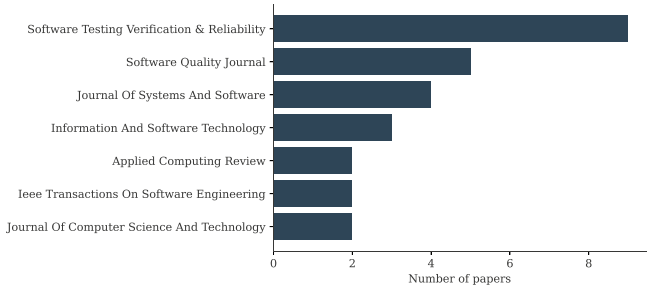


Fig. 6. Most important journals and number of papers per journal four our subset of extracted papers.

most important journals are *"Software Testing Verification & Reliability"*, *"Software Quality Journal"* and *"Journal of Systems and Software"*. I will give a short overview of these three journals.

Software Testing Verification & Reliability,[1] published by Wiley & Sons, is an international journal on theoretical and practical issues of software testing, verification and reliability and has an impact factor of 1.75 (2022). It publishes 8 issues per year. In the selection of our extracted papers, nine papers were published by this journal. *The Software Quality Journal*[2] is published by Springer. It publishes papers addressing quality management and effective construction of software systems. It offers methods, tools and products to measure and achieve better quality. It has an impact factor of 1.9 (2022) and a five-year impact factor of 2.1 (2022). It publishes four issues per year. *Journal Of Systems And Software*[3] is published by Elsevier. It covers general aspects of software engineering with a broad scope. The journal publishes research papers, state-of-the-art surveys, and reports of practical experience. It has an impact factor of 2.8 (2022) and publishes 12 issues per year.

Regarding the conferences, 11 of the 34 papers (32%) came from the International Conference on Software Engineering (ICSE), 6 conference papers (17%) were presented at the International Conference on Automated Software Engineering (ASE) and five papers (14%) were presented at the Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering.

The three journals "Software Testing Verification & Reliability", "Software Quality Journal" and "Journal of Systems and Software" published almost 50 percent of the extracted journal papers. The most important conference regarding the papers was the "International Conference on Software Engineering".

[1] https://onlinelibrary.wiley.com/journal/10991688
[2] https://www.springer.com/journal/11219
[3] https://www.sciencedirect.com/journal/journal-of-systems-and-software

4.2. RQ 2 — Problems, testing techniques and tools

In this section we want to discuss the results that answer what the research was about and which problems are the most important in recent years, which techniques have been used and what tools have been developed to resolve the problems.

4.2.1. RQ 2.1 Which problems are being addressed?

Ricca et al. (2019) identified three main problems related with web application testing, namely the *Fragility Problem*, that describes fragile test cases because of web element locators, that become invalid as web applications evolve. The second problem is named *Strong Coupling and Low Cohesion*. It describes that test cases often tend to be strongly bound to the implementation of the web page under test. The third problem is the *Incompleteness Problem*. It describes, that it is almost impossible to test all paths and the complete possible input space, even with automated tests. We can find these problems in our selection of extracted papers too.

We can identify 5 main problems with Web application testing, that are addressed by more than one of the extracted papers. They are given in Table 3. The papers were assigned to the questions that fit mostly to the general direction of the paper, even if multiple objectives were given. If no question fit the direction of the paper, we chose a new category of question. We will summarize these questions and possible answers, given by the relevant papers.

How to generate test cases and test models? We found 22 papers that dealt with the general question how to generate test cases, either automatically or make it easier to generate test cases, since programming test cases by hand is a time-consuming task. Here the incompleteness problem is tried to cope with by generating tests automatically.

One way to generate test paths is to use some kind of user aided method. Long et al. (2022) generate tests with a record/replay GUI. Here, a user navigates through a system and tests are generated from these paths. Test paths can also be obtained by using user logs as done by Sprenkle et al. (2013) to generate abstract test cases. Qi et al. (2019) use a keyword guided exploration to generate test models. Yousaf et al. (2019) generate tests from a flow model language models, that describes states and transitions. Bures et al. (2018) support exploratory testing using model reconstruction. Thummalapenta et al. (2013b) maps given tests to a graph build by crawling and Fard et al. (2014) uses existing tests and improves them through crawling. The advantages for this kind of test generation are, that specific, important paths can be chosen to generate tests, with the drawback, that the generation still is a manual job.

Another possibility is to generate tests from explored paths automatically. Here, a crawler analyzes the structure of the system and tests can be derived from a finite state model. Wang et al. (2016) uses this approach and generates a navigational graph to generate test cases. Biagiola et al. (2019b) generate tests from models with most diverse paths.

Other papers in this category have varied approaches. Stocco et al. (2017) generate page objects automatically to generate test cases easier. Qi et al. (2017) use pairwise testing to cover all combinations of parameters to achieve full form test and increase branch coverage for testing. Andrews et al. (2019) use a finite state machine to generate regression tests from fail-safe behavior. Boukhris et al. (2017) also generate tests from fail-safe behavior but use a genetic algorithm to generate failure scenarios. Matiello and Endo (Mattiello and Endo, 2022) infer a model from existing tests and generate tests from it. Leotta et al. (2018) developed a tool to migrate DOM based tests to a visual approach. Wang et al. (2023) generate test cases in parallel with genetic algorithms. Panda and Mohapatra (2021) prioritize and weight test scenarios based on user requirements.

Existing tests do not longer match the system under test (Regression): Here papers dealt with problems of regression testing. Sometimes the locators for a specific web element change in the system

Table 3
Questions that the extracted papers address.

Question	Num	Papers
How to generate test cases and test models?	23	Stocco et al. (2017), Long et al. (2022), Qi et al. (2017), Andrews et al. (2019), Boukhris et al. (2017), Yousaf et al. (2019), Bures et al. (2018), Sprenkle et al. (2013), Qi et al. (2019) Mattiello and Endo (2022), Leotta et al. (2018), Wang et al. (2016), Wang et al. (2023), Zou et al. (2014), Schur et al. (2013), Fard et al. (2014), Mariani et al. (2014), Clerissi et al. (2020) Thummalapenta et al. (2013b), Thummalapenta et al. (2013a), Bajaj et al. (2015), Biagiola et al. (2019b), Yandrapally et al. (2015)
What to do against breaking existing tests?	10	Leotta et al. (2016), Leotta et al. (2021), Nass et al. (2023), Imtiaz et al. (2021), Nguyen et al. (2021), Long et al. (2020), Qi et al. (2023), Stocco et al. (2018), Hammoudi et al. (2016), Nguyen et al. (2014)
How to measure the efficiency of tests?	9	Do and Hossain (2014), Nguyen et al. (2019), Mirshokraie et al. (2015), Leotta et al. (2023), Sherin et al. (2021), Habibi and Mirian-Hosseinabadi (2015), Eda and Do (2019), Mohd-Shafie et al. (2020), Mirzaaghaei and Mesbah (2014)
How to crawl more efficiently?	8	Abadeh (2021), Choudhary et al. (2014), Leithner and Simos (2021), Corazza et al. (2021), Guarnieri et al. (2017), Zheng et al. (2021), Yandrapally et al. (2020), Sherin et al. (2023)
How to detect errors in Javascript?	9	Ocariza et al. (2016), Jensen et al. (2013), Sung et al. (2016), Adamsen et al. (2018), Fard et al. (2015), Maezawa et al. (2013), Zhang and Wang (2017), Ocariza et al. (2017), Li et al. (2014a)
Is the application keyboard friendly?	2	Chiou et al. (2023), Chiou et al. (2021)
Are there cross browser compatibility problems?	2	Choudhary et al. (2013), Kwon et al. (2018)
We need test applications to validate methods.	1	Soto-Sánchez et al. (2022)
How to repair I18n errors?	1	Mahajan et al. (2021)
How to test search engines?	1	Zhou et al. (2016)
How to write tests faster?	1	Bünder and Kuchen (2019)
How to make exploration testing easier?	1	Leveau et al. (2020)
Which tests to run first?	2	Biagiola et al. (2019a), Khanna et al. (2019)
What are the reasons for flaky tests?	1	Romano et al. (2021)

under test, because the element has been moved or another element was added before. With that, the location in the DOM will change and as a result, a test case will fail, if the locator is not flexible enough. This is the fragility problem mentioned before. Papers in this category search for better locators to deal with this problem. Leotta et al. (2016) developed Robula+, an algorithm for robust XPath locators. Later they also developed a tool Sidereal for robust XPath locators (Leotta et al., 2021). Nass et al. (2023) propose a similarity based XPath locator to repair test suites. Imtiaz et al. (2021) give a method to repair test suites by DOM comparison with earlier versions of the application. Nguyen et al. (2021) use the semantic structure of web pages to construct XPath locators. Stocco et al. (2018) use computer vision algorithms to find locators visually.

How to measure the efficiency of tests? In this category, the papers look for different ways to test the efficiency of tests, either by mutation testing or by looking at different metrics like code coverage or path coverage. Efficiency here has two meanings. It is important to identify how many errors can be found in a certain area. On the other hand, it is equally crucial to know how much of the system under test is covered by the tests. Both are important to prioritize tests that should be run first, since testing the complete test framework is too cost-intensive. More efficient tests should be preferred depending on the context (for example which parts of the code are changed).

The first category of papers uses mutation testing to evaluate the quality of tests. Habibi and Mirian-Hosseinabadi (2015) use graph based mutation testing to find errors in event driven test graphs. Mirshokraie et al. (2015) improve mutation testing to evaluate tests. Leotta et al. (2023) also use a mutation testing framework for better test evaluation. Mutation testing is time-consuming, since small changes have to be made in the source code to check if tests can detect these changes. An advantage of this kind of testing is, that one can estimate the quality of the test system.

Another possibility is to use some kind of coverage criteria to see which tests cover most of the system. Sherin et al. (2021) compare and evaluate coverage criteria for automatic testing techniques. Nguyen et al. (2019) select test cases based on output coverage metrics. Do and Hossain (2014) checks the impact of changed source code to test paths generated by test in the system. Eda and Do (2019) execute regression tests only for code paths that have actually changed. Mohd-Shafie et al. (2020) priorities test cases through changed source code.

Both mutation testing methods and methods based on coverage give opportunities to estimate the quality of a test framework to increase the efficiency.

How to crawl more efficiently? These papers use crawling techniques to generate a graph model. Choudhary et al. (2014) use different techniques to solve common problems while crawling like recurring menu links. Zheng et al. (2021) and Sherin et al. (2023) use reinforcement learning to generate models of higher quality. Yandrapally et al. (2020) and Corazza et al. (2021) find near-page duplicates in the models. Guarnieri et al. (2017) propose a method to reset applications between test by using checkpoints to accelerate the testing process.

How to detect errors in Javascript? Here we find some papers that use static Javascript analysis like Ocariza et al. (2017) and Sung et al. (2016). Adamsen et al. (2018) and Zhang and Wang (2017) try to find race conditions in applications that use Ajax.

> We could identify the most important questions that the extracted papers try to answer. More than 30% of the papers gave answers on how to generate test cases and test models. 15% of the papers were about breaking tests.

4.2.2. RQ 2.2 : Testmodel

Doğan et al. (2014) differentiate between three main test models that we also want to use in this study. These models were either used

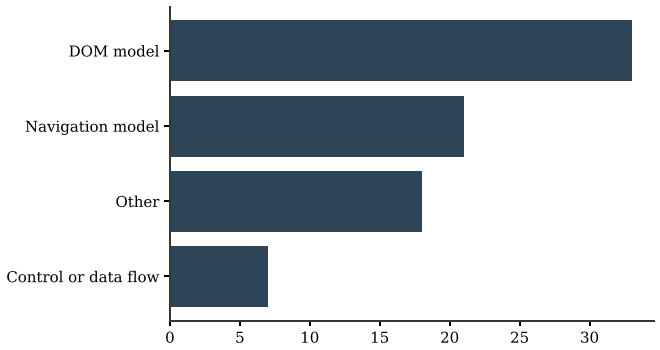


Fig. 7. Test models used in extracted papers. Some of the papers used more than one model.

as input for testing or the models were inferred from the existing system under test. The observed test models that recurred several times where:

- Navigation models: models that describe the flow through the system under test, such as finite-state machines (FSM). They usually describe pages as states or vertices and events like clicks as edges.
- Control or data flow models: These models usually describe the data flow as instructions in modules or functions.
- DOM models: use the Document Object Model (DOM) of web pages as source for the test model.
- Other: models that do not fit the description above or papers that do not use a test model at all.

Navigation models, usually modeled as finite state machines, describe pages or parts of web pages as states or vertices and links or Javascript actions as edges between these vertices. These models are usually very flexible and can be adjusted easily if changes occur in the system under test. It is also very easy to traverse these networks or find paths for testing.

Control or data flow models describe the test model as data objects with functions to work on these data. To test these data models, usually scripts are written to describe a test through sequences of consecutive steps.

DOM test models use the DOM specifically to test web applications, *e.g.* by changing or analyzing the DOM. Especially Javascript is tightly connected to the DOM, since it can manipulate it directly.

As shown in Fig. 7, DOM based models are used by most of the papers. This is on the one side a result from many papers that work on Javascript problems and work exclusive on the DOM, on the other side depend crawler based solutions also often on the DOM for analysis and blend navigational models with it. These test solutions are easy to generalize to other systems, since the DOM can be inferred directly from the browser and a navigational model can be build dynamically, while control or data flow models are more customized to the system under test (see Table 4).

Most studies use the DOM as test models for their test suites.

4.2.3. RQ 2.3 : Fault models

The fault models used in the extracted papers describe what kind of errors the paper was focused on. The categories for the fault models were found by grouping the papers that addressed the same errors and infer matching categories.

Table 5 shows the different error models found in the extracted papers. Most of the papers deal with general problems of navigational or functional errors in web applications. All the papers that work on model building and general web testing fall into this category. This includes papers that work with graph models as well as existing test scripts that simulate user interactions.

Broken locators are a problem discussed in many papers. These problems usually arise with existing test cases. When web applications change in new versions, the locators (*e.g.* X-paths) to HTML elements like buttons or text inputs in existing tests do not match anymore. Even though these errors are a subset of navigational or functional errors, we chose to name them separately here to highlight the importance of these problems.

Another group of papers dealt with Javascript errors. Some papers do static Javascript analysis or test in other ways on the client side. Closely related to this fault model are problems with asynchronous communication, since they are usually triggered from Javascript. Since Javascript runs asynchronously, there is a danger of race conditions. An advantage for static analysis or checks for race conditions is that applications can be checked without existing test cases as implicit oracles.

We found two papers that checked specifically for problems with keyboard accessibility and how difficult it is to use the web application without a mouse. Two papers analyzed problems with cross browser compatibility, a problem where an application behaves differently in different browsers.

We identified and classified some fault models. Most of the papers are about navigational or functional testing of web pages in general, but we found a considerable amount of papers targeting specifically the problem of broken locators and some papers on Javascript testing.

4.2.4. RQ 2.4 : Tools

Table 6 shows the public accessible tools that are used in the extracted papers. 47 of the 72 extracted papers (65%) mentioned tools. We tried to find, if they are publicly available, either from a mention in the paper itself or by searching the internet manually. We found only 15 of them have a link that is publicly accessible, that means only 32% of the tools are available. Even though the numbers are too small to be statistically relevant, it can be mentioned, that in the paper by Doğan et al. (2014) only 21% of the tools were available for download. This points to a positive trend.

There are some open-accessible tools that try to solve the problem of broken locators. **Similo** is a tool developed by Nass et al. (2023) that uses similarity-based XPath locators to find items in the DOM in the case of changed XPaths. With this change XPaths can be repaired. **Robula+** by Leotta et al. (2016) is in the same category. It is a GitHub repository with an algorithm to generate robust XPath locators. **Vista** is a tool developed by Stocco et al. (2018). It uses computer vision algorithms to find elements with locators that are not correct anymore to repair broken tests. Then there are tools to detect Javascript faults. **Autoflox** detects Javascript errors in code that tries to access DOM elements that are not existent, which results in null exceptions. **Mutandis** by Mirshokraie et al. (2015) uses mutation testing to find errors in Javascript. **AjaxRaver** find race conditions in Javascript code that loads content with Ajax. **ComFIx** generates DOM fixtures for Javascript testing by symbolic analysis of the code while executing concrete paths.

The other tools do not fit into the first categories and are for general web testing. **Tapir** is a regression testing framework developed by Bures et al. (2018). **MoLeWe** (Model-based testing Leverage for Web), developed by Mattiello and Endo (2022) is a tool that supports test case generations. It presents a graphical user interface to a model of a web application to generate test cases. **Qexplore** by Sherin et al. (2023) is a dynamic exploration tool for web applications that uses Q-learning and builds a state-flow graph for test case generation. **Apogen** (Automatic Page Object Generator) by Stocco et al. (2017) created Java page objects from web applications. A crawler scans the application, and Java code for page objects is generated, that can be used for testing.

Table 4
Testmodels used in extracted papers.

Testmodel	Papers
DOM model	Leotta et al. (2018), Leithner and Simos (2021), Long et al. (2022), Nguyen et al. (2021), Leotta et al. (2021), Leotta et al. (2016), Nass et al. (2023), Long et al. (2020), Corazza et al. (2021), Qi et al. (2023), Zou et al. (2014), Li et al. (2014a), Mirzaaghaei and Mesbah (2014), Sung et al. (2016), Fard et al. (2014), Stocco et al. (2018), Adamsen et al. (2018), Chiou et al. (2021) Hammoudi et al. (2016), Choudhary et al. (2013), Mariani et al. (2014), Kwon et al. (2018), Clerissi et al. (2020), Zheng et al. (2021), Thummalapenta et al. (2013b), Fard et al. (2015), Thummalapenta et al. (2013a), Maezawa et al. (2013), Zhang and Wang (2017), Chiou et al. (2023), Ocariza et al. (2017), Yandrapally et al. (2020), Bajaj et al. (2015), Yandrapally et al. (2015)
Navigation model	Bures et al. (2018), Sprenkle et al. (2013), Qi et al. (2019), Wang et al. (2016), Wang et al. (2023), Sherin et al. (2023), Habibi and Mirian-Hosseinabadi (2015), Qi et al. (2017), Andrews et al. (2019), Choudhary et al. (2014), Schur et al. (2013), Fard et al. (2014), Biagiola et al. (2019b), Biagiola et al. (2019a), Chiou et al. (2021), Zheng et al. (2021) Thummalapenta et al. (2013b), Thummalapenta et al. (2013a), Maezawa et al. (2013), Chiou et al. (2023), Yandrapally et al. (2015)
Control or data flow	Mattiello and Endo (2022), Nguyen et al. (2019), Mohd-Shafie et al. (2020), Bünder and Kuchen (2019), Stocco et al. (2017), Boukhris et al. (2017), Guarnieri et al. (2017)
Others	Leveau et al. (2020), Eda and Do (2019), Imtiaz et al. (2021), Mahajan et al. (2021), Sherin et al. (2021), Ocariza et al. (2016), Do and Hossain (2014), Soto-Sánchez et al. (2022), Panda and Mohapatra (2021), Mirshokraie et al. (2015), Leotta et al. (2023), Abadeh (2021), Yousaf et al. (2019), Zhou et al. (2016), Khanna et al. (2019), Jensen et al. (2013), Nguyen et al. (2014), Li et al. (2014a), Romano et al. (2021)

Table 5
Fault models used in extracted papers.

Testmodel	Number	Papers
Navigation/Functional	31	Bures et al. (2018), Sprenkle et al. (2013), Qi et al. (2019), Mattiello and Endo (2022), Leveau et al. (2020), Wang et al. (2016), Wang et al. (2023), Sherin et al. (2023), Sherin et al. (2021), Bünder and Kuchen (2019), Do and Hossain (2014), Leotta et al. (2023) Habibi and Mirian-Hosseinabadi (2015), Stocco et al. (2017), Qi et al. (2017), Andrews et al. (2019) Boukhris et al. (2017), Leithner and Simos (2021), Abadeh (2021), Yousaf et al. (2019), Choudhary et al. (2014), Guarnieri et al. (2017), Mirzaaghaei and Mesbah (2014), Fard et al. (2014) Biagiola et al. (2019b), Mariani et al. (2014), Clerissi et al. (2020), Zheng et al. (2021), Thummalapenta et al. (2013b), Thummalapenta et al. (2013a), Maezawa et al. (2013)
Broken Locators	11	Nass et al. (2023), Leotta et al. (2016), Imtiaz et al. (2021), Leotta et al. (2021), Nguyen et al. (2021), Leotta et al. (2018), Long et al. (2020), Qi et al. (2023), Stocco et al. (2018), Hammoudi et al. (2016), Bajaj et al. (2015)
Javascript errors	7	Ocariza et al. (2016), Mirshokraie et al. (2015), Long et al. (2022), Li et al. (2014a), Sung et al. (2016), Fard et al. (2015), Ocariza et al. (2017)
Asynchronous communication	3	Jensen et al. (2013), Adamsen et al. (2018), Zhang and Wang (2017)
Duplicate states	2	Corazza et al. (2021), Yandrapally et al. (2020)
Keyboard accessibility	2	Chiou et al. (2021), Chiou et al. (2023)
Cross browser compatibility	2	Choudhary et al. (2013), Kwon et al. (2018)
Others	5	Zhou et al. (2016), Zou et al. (2014), Biagiola et al. (2019a), Romano et al. (2021), Yandrapally et al. (2015)

WbCheck is a tool that helps to set checkpoints under testing. When running tests, usually databases and sessions have to be reset. This tool aims to help with this problem. **DIG** is a tool by Biagiola et al. (2019b) where a state flow graph is built, and test cases are generated on that model, by using a distance measure between test cases (inputs that consist of action, inputs, fields). The most diverse test cases are executed first, and test coverage is increasing faster. **TEDD** is a tool by Biagiola et al. (2019a) that finds dependencies of tests with NLP from test descriptions (*e.g.* creating an object, reading the object).

Fig. 8 shows the tools per year. There are more tools not publicly accessible than tools with public access. The numbers of tools are unfortunately too small to find a trend.

Table 6
Tools that are public accessible used in the extracted papers.

Toolname	Ref	URL	Technology	Comment
Similo	Nass et al. (2023)	https://github.com/michelnass/Similo2	Java	Similarity based XPath locators
Robula+	Leotta et al. (2016)	https://github.com/cyluxx/robula-plus	NodeJS	Robust XPath-based locators
Vista	Stocco et al. (2018)	https://github.com/saltlab/vista	Java	Broken locators are repaired through visual analysis.
Autoflox	Ocariza et al. (2016)	https://people.ece.ubc.ca/frolino/projects/autoflox/	Javascript	find DOM-related Javascript faults
Mutandis	Mirshokraie et al. (2015)	https://github.com/saltlab/mutandis/	Java	Javascript mutation testing tool
AjaxRacer	Adamsen et al. (2018)	https://github.com/cs-au-dk/ajaxracer	Javascript	Checks Javascript code for race conditions
ConFix	Fard et al. (2015)	https://github.com/saltlab/Confix	Java	A concolic DOM fixture generator for Javascript
Tapir	Bures et al. (2018)	https://www.tapir-test.io/	Java	A complete regression testing framework
MoLeWe	Mattiello and Endo (2022)	https://github.com/guimattiello/MoLeWe	Java	An eclipse plugin generating test paths
Qexplore	Sherin et al. (2023)	https://github.com/salmansherin/QExplore	Python	Generates a state-flow model from crawling exploration
Apogen	Stocco et al. (2017)	https://github.com/tsigalko18/apogen	Java	Generation of page objects
WebCheck	Guarnieri et al. (2017)	https://infsec.ethz.ch/research/software/webcheck.html	Python	Setting test execution checkpoints
Testilizer	Fard et al. (2014)	http://salt.ece.ubc.ca/software/testilizer	Java	Existing tests are improved by crawling.
DIG	Biagiola et al. (2019b)	https://github.com/matteobiagiola/FSE19-submission-material-DIG	Java	Diversity test path priorization
TEDD	Biagiola et al. (2019a)	https://github.com/matteobiagiola/FSE19-submission-material-TEDD	Java	Test dependency detection from test descriptions.

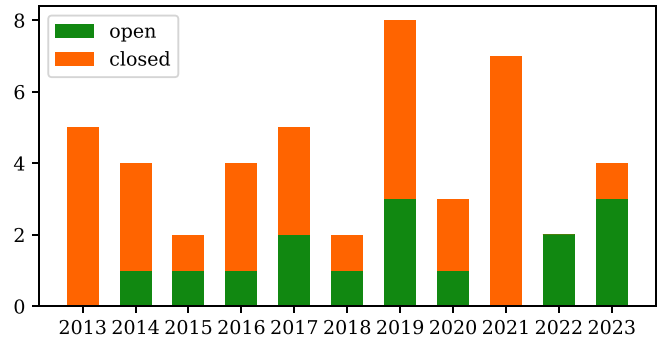


Fig. 8. Tools developed and used in the selected papers, both public available and closed source or not available.

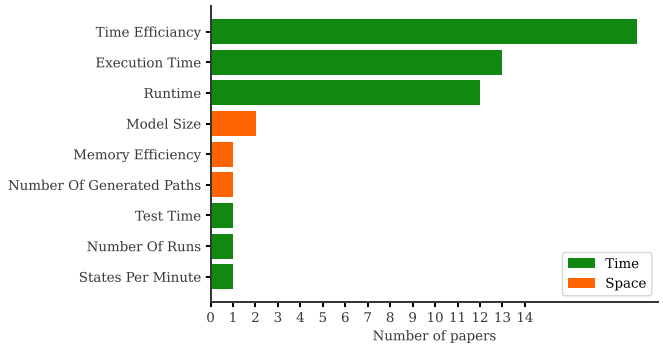


Fig. 9. Cost metrics for the selected papers.

It should be mentioned that most of the tools are given as source code in a Github repository or in a similar form. Only Tapir is a testing framework that has been transformed to a professional tool with a licensing model.

> We found 15 of the 47 developed tools open accessible. This is more than 30 percent and a higher percentage than 10 years ago, but most of the tools in the extracted papers are still not public accessible.

4.3. RQ 3 — Empirical studies

In these sections we will examine the empirical studies of the extracted papers. The metrics for cost and effectiveness should give an opportunity to compare different studies and are therefore one indicator for the quality of a paper. Describing the threats to validity in depth is another indicator for high quality research. The last research questions look at the level of rigor and industrial relevance of the examined papers.

4.3.1. RQ 3.1 : metrics for cost and effectiveness

One of the important factors for a study is the assessment of quality of the tool, that has been developed or used by the authors. Metrics for cost and effectiveness are possible metrics for such an assessment, that enable comparison with other tools. Coppola and Alégroth (2022) give an overview of different metrics used in papers about web application testing. They compiled a taxonomy of 55 metrics, distributed in 17 categories, and 4 higher level categories. In general, they found more mentions of Model and Code level metrics than Functional and GUI level metrics. This is also reflected in our findings.

Fig. 9 shows the cost metrics found in the selected papers. Most papers use some type of time metric to evaluate the costs. However, two papers use the model size as a cost metric (Sprenkle et al., 2013) and (Qi et al., 2019), one paper uses the number of generated paths (Do and Hossain, 2014) and Wang et al. (2016) uses memory efficiency in addition to time efficiency as a cost metric.

Regarding the efficiency metrics, we have a widespread result of multiple metrics used. The results are shown in Table 10. We have classified the different metrics in four groups:

- coverage
- fault detection
- time efficiency
- comfort

Most of the papers use some kind of coverage metric. In some cases, this means literally code coverage, in other cases it can be coverage of existing test models, DOM coverage or the number of identified elements in a web page. Coverage alone gives only an indication of how much testing of a system has been done. Fault detection metrics count the number of detected errors, to give a measure of quality of the tool. The other two categories are time efficiency in cases where time is an efficiency indicator. Comfort is a metric in Bünder and Kuchen (2019) where tests have to be written with the help of a tool.

Fig. 10 shows the number of papers that used the different efficiency metrics categories.

> More than 30 percent of the papers use a time efficiency metric to measure cost. For efficiency measurement, more than 50 percent of the papers use coverage metrics, 30 percent of the papers use a measurement of fault detection.

4.3.2. RQ 3.2 : Analysis of threats to Validity

Fig. 11 shows a ranking of the most important threats to validity found in the extracted papers. The most important threat for validity in the extracted papers is the problem of testing new methods only on

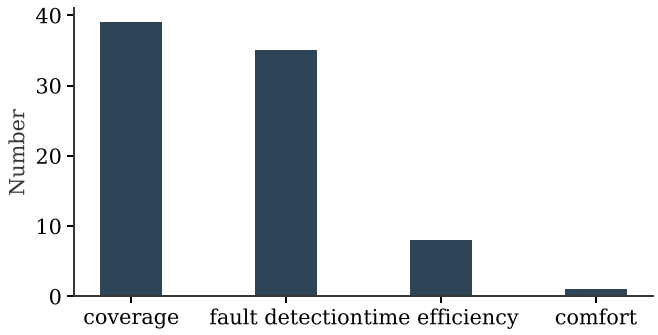


Fig. 10. Number of papers that used efficiency metrics classes.

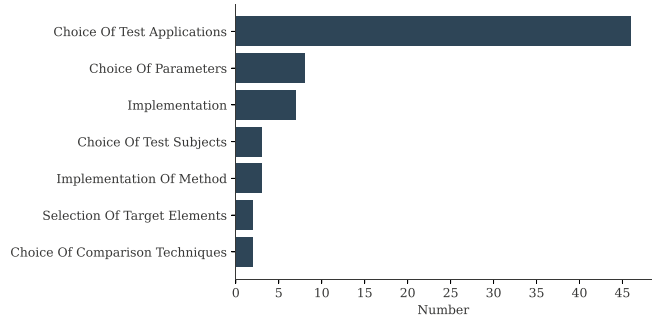


Fig. 11. A ranking of threats for validity in the extracted papers.

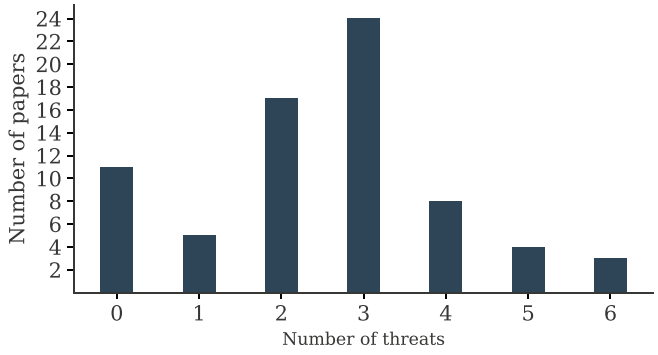


Fig. 12. Frequency of papers with a certain number of threats for validity.

some chosen test applications. When looking at the data for RQ 5 about datasets and validation, this becomes more apparent.

61 of the 72 papers (85%) mention threats to validity. In addition, most papers mention both internal and external threats. Fig. 12 shows the frequency of the number of papers per number of threats of validity mentioned. We can see in this figure that more than half of the extracted papers have three or more threats discussed.

Almost all papers mention threats for validity. This has become a standard for quality papers. The most mentioned threat for validity was the choice of test applications that has been used for validation.

4.3.3. RQ 3.3 : Rigor and Relevance

The level of rigor and industrial relevance is an indicator for the quality and applicability of a study. To analyze both qualities, it is important to quantify these properties. Ivarsson and Gorschek (2011) used a method to quantify rigor and industrial relevance, and we used their method and categories for our assessment.

For the level of rigor, three categories have been used to be assessed each with a score of 0 for a weak description of rigor in the category, 0.5 for a medium description or 1 for a strong description. The categories are (1) context described, (2) study design described (*e.g.* variables measured, used control) and (3) validity discussed.

Table 7 shows an overview of these categories For the described context, we evaluated how well the reader could understand the context of the study (*e.g.* development mode, development speed, company maturity). The other points were given regarding how well the study design was described and if the threats for validity were discussed.

Fig. 13 shows the results for the three categories. Validity is discussed in almost every paper we have examined. This is a good finding for quality, as it is important to reflect the boundaries and possible weakness of a study. Both the context and the study design are described well enough in nearly all papers, so that the reader can understand them.

Industrial relevance was measured in four categories:

1. Subjects: The subjects in the study are representatives of the intended users of the technology.
2. Context: The evaluation is performed in a setting representative of the intended usage setting.
3. Scale: The scale of the applications used for evaluation is of realistic size.
4. Research method: The mentioned methods investigate real world situations (*e.g.* case studies)

Each category was either scored with 1 if the category was applicable to the study or 0 otherwise. All of our examined papers were scored in this way.

Fig. 14 shows the results for each category. The category for Subjects has a low score for almost all papers (Fig. 14(a)). Only few papers used subjects for their studies that came from the field with enough experience, if manual testers were needed at all. Since methods are used, that automatically test the systems, this category was not applicable for all examined papers. When manual testing was done, usually students were used without real world experience. The same can be said about industrial context and research methods (Figs. 14(b) and 14(c)). Usually, experiments were conducted at the university and not in an industry setting. The research methods to evaluate the results were in the context of laboratory experiments and not in real world situations.

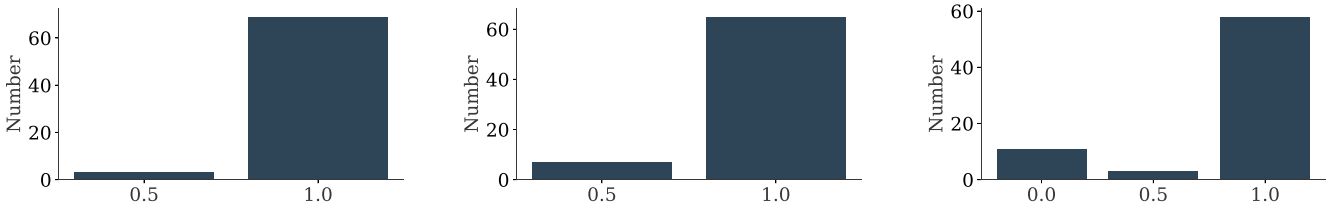
As for the scale of test applications, we found a little bit more positive results (Fig. 14(d)). We gave a score of 1 if the test applications had a size larger than 10,000 LOC, while we gave a 0 if the number of LOC was lower than 10,000. Here we used the same number as Doğan et al. (2014), for comparison. This number cannot express the complexity of an application, but is a measurable variable to distinguish between toy examples and big real world applications. Even though most of the studies used applications that were not used in the industry, the size of some of the test applications were comparable with real world applications in more than 60% of the papers. Only 6 papers used web applications from industry (Leveau et al., 2020; Mahajan et al., 2021; Zhou et al., 2016; Long et al., 2020; Thummalapenta et al., 2013a; Nass et al., 2023). We will take a closer look at validation applications in RQ 5.

We can summarize that most of the examined papers have in general little industrial relevance, since experiments usually are conducted in a university or in laboratory settings without involvement of the industry. Only the size of test applications shows that the examined methods would be applicable in real world scenarios. Further research is necessary to find the impact of newer software testing techniques in the industry.

Fig. 15 shows the connection between the values for rigor and for relevance. Here the numbers for relevance are added up to a sum as well as the numbers for rigor. It shows that the papers with the highest level of industrial relevance also have the highest level of rigor. Papers

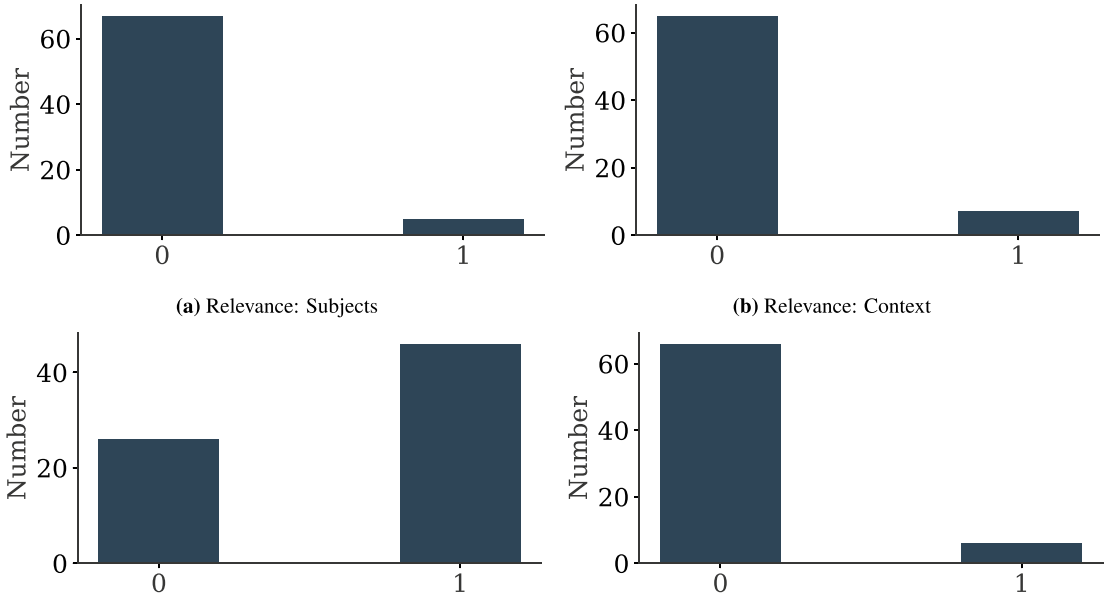
Table 7
Scoring for rigor evaluation as in Ivarsson and Gorschek (2011). Each aspect would be scored and added to a summed value for rigor for each paper.

Aspect	Strong description (1)	Medium description (0.5)	Weak description (0)
Context described	The context is described in a degree where a reader can understand and compare the described context to another context.	The context is mentioned but not described to the degree that the reader can understand and compare it to another context.	The context is not described.
Study design described	The study design is described in a degree where a reader can understand it.	The study design in briefly mentioned.	No description of the design of the evaluation.
Validity discussed	The validity is discussed in detail.	The validity is mentioned, but not described in detail.	No description of any threats to validity



(a) Context described **(b)** Study design described **(c)** Validity discussed

Fig. 13. Detailed figures for rigor.



(a) Relevance: Subjects **(b)** Relevance: Context

(c) Relevance: Scale **(d)** Relevance: Research method

Fig. 14. Detailed figures for industrial relevance.

with both high relevance and rigor are Sprenkle et al. (2013), Leotta et al. (2016) and Imtiaz et al. (2021).

For the rigor of the examined paper, we found positive results. Most papers had a good description of context and study design and the mentioning of threats for validity in almost all papers is a good sign. Most papers have little industrial relevance, since industry is not involved in most studies.

4.4. RQ 4 — Empirical evidence

We want to look at empirical evidence with three research questions. Evidence for scalability, comparison of different techniques and evidence for the different techniques itself.

4.4.1. RQ 4.1 : Evidence regarding scalability

Scalability is discussed in only 4 papers. Boukhris et al. (2017) discusses scalability in connection with building failure mitigation models for web application. A genetic algorithm generates failure scenarios, and from there, failure mitigation test paths are transformed into an executable test suite. Scalability is in this context examined as its own research question and proved for a test application with a bigger size, but not examined in detail and how it behaves for different test application sizes.

Andrews et al. (2019) use a similar approach to generate regression test paths with help of FSMWeb — a finite state machine. They do not use a genetic algorithm, but use coverage criteria to generate failure scenarios. Scalability is mentioned in this context, but not analyzed deeper.

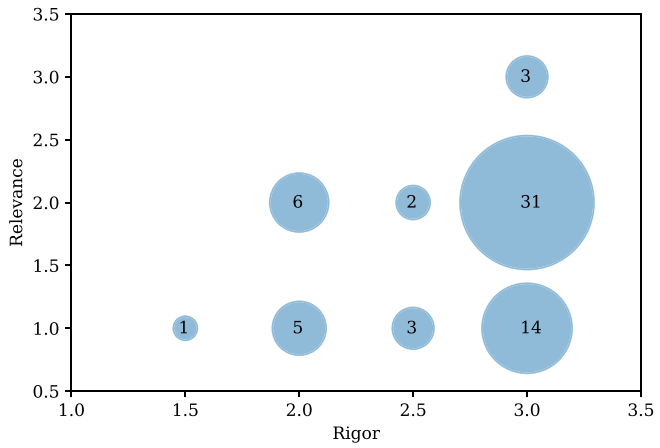


Fig. 15. Rigor and Relevance.

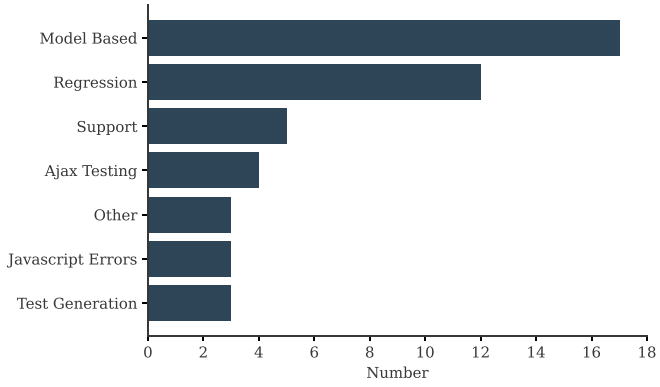


Fig. 16. Themes used in extracted papers.

Soto-Sánchez et al. (2022) confirm scalability as a problem in web application testing, and offer a dataset of test applications with regression bugs and corresponding fixes, but do not go in detail further.

Zheng et al. (2021) developed a tool "WebExplor", that uses reinforcement learning to crawl a web application and generates a state flow model. They tested on the top 50 most popular web applications (ranked by Alexa) to show scalability.

As a conclusion for this research question, scalability is not discussed very much. Doğan et al. (2014) had also only three papers discussing it. So we can say that the situation has not changed since 2013. The reasons may be that scalability is in reality not such a big problem after all, or that there is simply a research gap that is not investigated enough.

> Scalability is a problem that is not sufficiently addressed in the extracted papers. Only 4 papers of 72 discuss scalability.

4.4.2. RQ 4.2 : Have different techniques been empirically compared with each other?

To compare different techniques, we have to get an overview about the different techniques used in the extracted papers first. We developed a set of theme areas for different web application techniques by following the guidelines in Cruzes and Dyba (2011) for thematic analyses. Fig. 16 shows the frequencies of the different technique themes.

Table 8 lists all the themes of technique and the associated papers. Since we allowed multiple themes, some of the papers appear several times in different theme categories.

For a detailed description of papers that compared techniques, we mention each paper just once even if they appear multiple times for different techniques, as long as the specific paper is not vital for the description. In total, we only found 12 papers that compared their results with other techniques.

Regression testing

A high number of papers deal with broken locators under regression testing. One of the most important papers is the work of Leotta et al. (2016). They build a tool 'Robula+', that can be seen as a baseline for robust locators. They compared their results with Montono (Montoto et al., 2011), the internal Selenium locators and FirePath, a Firefox add-on.

Using OpenCV, Stocco et al. (2018) analyze web pages visually to find broken locators and repair them with their tool 'Vista'. They compare their results with 'WATER' (Choudhary et al., 2011)

Long et al. (2020) use the tool 'WebRR' with a record and replay approach to generate multiple locators to repair possible broken locators. They compared their results with Vista (Stocco et al., 2018).

Leotta et al. (2021) compared their results for Sidereal a new tool, that uses a statistical adaptive algorithm with their earlier results for Robula+ and the results from Montoto Montoto et al. (2011).

Nguyen et al. (2021) built locators based on semantic structures of web pages by looking at the neighbor elements of a locator element. They compared their results with Robula+ (Leotta et al., 2016). Nass et al. (2023) together with Leotta as coauthor based their work on Robula+. Their tool "Similo", that use a weighted similarity score of neighboring web elements had also better results than Robula+. Qi et al. (2023) developed the tool 'Semter' (Semantic Test Repair) to capture relevant semantic information from test execution and calculate semantic similarity to repair broken locators. The results are compared with Vista, WATER and WebEVO.

Imtiaz et al. (2021) try to repair broken locators in Capture-Replay test scripts. This is a slightly different approach. They compare the DOM structure between different versions and try to find similar web elements for broken or missing locators. They compare their results with WATER (Choudhary et al., 2011).

Test generation

Studies in this category generate test cases automatically. Qi et al. (2017) generates tests for forms by exploring the state space with an FSM and finding parameters and constraints in a form from there. They compared their results for their tool ComJaxTest with the tools Crawljax (Mesbah et al., 2012) and VeriWeb (Benedikt et al., 2002). Another tool by Qi et al. (2019), called 'KeyjaxTest' proposes a keyword guided exploration strategy for test model construction. They compare the efficiency with FeedEx (Fard and Mesbah, 2013), Depth-First and Breadth-first algorithms. Bünder and Kuchen (2019) introduce a specification language to generate automatically test cases from BDD like feature descriptions. They compare their results for the tool Slang with JBehave (JBehave, 2024).

The tool DIG by Biagiola et al. (2019b) generates a state flow graph. Test cases are generated on that model, by using a distance measure between test cases. The most diverse test cases are executed first. By increasing diversity in the test paths, they increase coverage faster. They compared their tool with SubWEB and ATUSA.

Model based

All of these techniques have in common that they develop or are dependent on a model of the web application. In many cases this is a graph model states and event-links between states. Tools like Crawljax are often used to generate these models.

The tool 'WATEG' by Thummalapenta et al. (2013b) scans the system with a crawler and builds a graph, then it searches the given tests to map them to the graph and run them on the model. In this way, the crawling is directed to given tests. They compare their results with KPATH+, an enhanced version of ATUSA.

Table 8
Technique themes and associated papers.

Theme	# of papers	Percentage	# comparisons	References
Model based	22	30%	8	Mattiello and Endo (2022), Wang et al. (2016), Long et al. (2022), Imtiaz et al. (2021), Sherin et al. (2023), Andrews et al. (2019), Boukhris et al. (2017), Leithner and Simos (2021), Abadeh (2021), Leotta et al. (2018), Yousaf et al. (2019), Choudhary et al. (2014), Schur et al. (2013), Zou et al. (2014), Fard et al. (2014), Zheng et al. (2021), Thummalapenta et al. (2013b), Thummalapenta et al. (2013a), Bajaj et al. (2015), Yandrapally et al. (2015) Andrews et al. (2019), Boukhris et al. (2017)
Regression	11	15%	9	Leotta et al. (2016), Nass et al. (2023), Nguyen et al. (2019), Leotta et al. (2021), Eda and Do (2019), Panda and Mohapatra (2021), Long et al. (2020), Qi et al. (2023) Nguyen et al. (2021), Stocco et al. (2018), Hammoudi et al. (2016)
Javascript errors	9	12%	2	Ocariza et al. (2016), Li et al. (2014a), Sung et al. (2016), Jensen et al. (2013), Adamsen et al. (2018), Fard et al. (2015), Maezawa et al. (2013), Zhang and Wang (2017), Ocariza et al. (2017)
Support	8	11%	0	Nguyen et al. (2019), Bünder and Kuchen (2019), Sherin et al. (2021), Soto-Sánchez et al. (2022), Bures et al. (2018), Leveau et al. (2020), Mariani et al. (2014), Clerissi et al. (2020)
Test case priorization	5	7%	1	Mohd-Shafie et al. (2020), Panda and Mohapatra (2021), Khanna et al. (2019), Biagiola et al. (2019a)
Exploration testing	3	4%	1	Nguyen et al. (2019), Leveau et al. (2020), Bures et al. (2018)
Mutation testing	3	4%	0	Mirshokraie et al. (2015), Leotta et al. (2023), Habibi and Mirian-Hosseinabadi (2015)
Test generation	5	7%	3	Stocco et al. (2017), Do and Hossain (2014), Qi et al. (2019), Qi et al. (2017), Biagiola et al. (2019b)
Duplicate states	2	3%	2	Corazza et al. (2021), Yandrapally et al. (2020)
Session based testing	1	1%	0	Sprenkle et al. (2013)
Other	11	15%	3	Zhou et al. (2016), Wang et al. (2023), Nguyen et al. (2014), Guarnieri et al. (2017), Mirzaaghaei and Mesbah (2014), Mahajan et al. (2021), Chiou et al. (2021), Chiou et al. (2023), Choudhary et al. (2013), Kwon et al. (2018), Romano et al. (2021)

With tool 'Testilizer' by Fard et al. (2014) existing tests are run while generating a State Flow Graph that saves actions and assertions, analyses the DOM and expands on the existing tests, thus generating a greater and better set of tests.

Andrews et al. (2019) mention the work of Boukhris et al. (2017) and Boukhris is even coauthor of the first paper. They propose a method for black box model based regression testing, to reduce the number of tests that have to be run. There is no direct comparison of techniques. Both authors compare their results with general techniques, not mentioned by a paper (*e.g.* exhaustive search, change of parameters).

Leithner and Simos (2021) uses also a crawling based exploration of the system under test. Their main goal is to find all states and events that lead to new states. They compares their results for their tool "CHIEv" with their own older tool "XIEv", but also CrawlJax (Mesbah et al., 2012), wget, Skipfish (Zalewski et al., 2024) and w3af (W3afOpenSource, 2024).

Abadeh (2021) use genetic algorithms to develop a minimum set of test suits to identify changes from existing test models. They compare their results with different techniques NSGA-II (Deb et al., 2002), GA_ANFIS (Anwar et al., 2019), GA_order (Konsaard and Ramingwong, 2015).

WebExplor by Zheng et al. (2021) uses CrawlJax to generate a model. Since these models can be complex and very big, reinforcement learning methods are used to explore the system and to generate action sequences satisfying temporal logical relations. They compare the results with DIG, SUBWEB and random exploration. Mattiello and Endo (2022) infer models with their tool MoLeWe from existing tests. This model can then be used to generate better test code. They compared their tool with web testing without the tool.

Exploratory Testing

Exploratory testing is usually done manually by a tester who explores the system. The advantage is, that unknown errors, and logical errors of an application can be detected more easily by a human. The drawback is of course that it is time consuming. Leveau et al. (2020) assist testers under exploratory testing by suggesting real time probabilities for next interactions with the system. They compare their results with manual exploration testing without their tool.

Javascript

These are techniques that are specialized for Javascript testing. Li et al. (2014a) present SymJS, a symbolic execution checker for Javascript. Web events are discovered and associated Javascript code is symbolically executed to check for errors and generate test cases. They compare the results with ARTEMIS (Artzi et al., 2011). Sung et al. (2016) developed JSDEP, a static DOM event analysis tool to find dependencies of javascript functions. They compared their results also with ARTEMIS.

Others

Mahajan et al. (2021) try to repair i18n errors. They compare their tool with an earlier version iFix (Mahajan et al., 2018).

29 of 72 papers (40%) compare their results with other techniques.

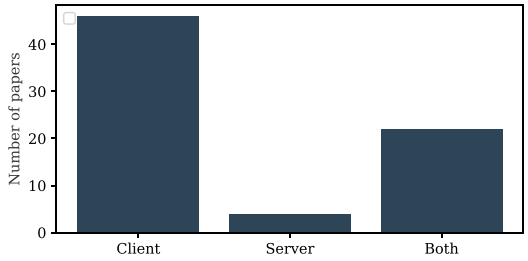


Fig. 17. Number of extracted papers that use techniques only on client side, only on server side or on both sides.

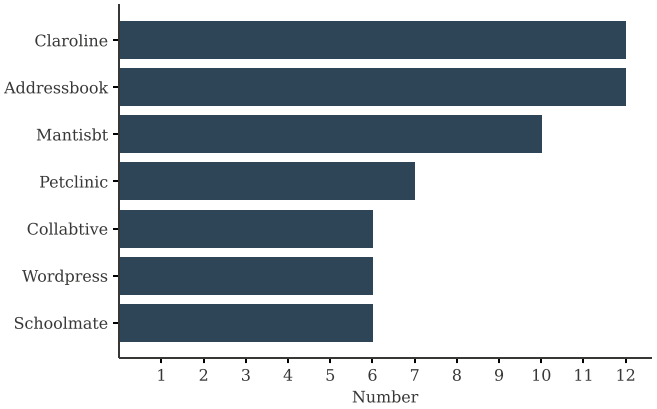


Fig. 18. Applications used for validation used by more than 5 papers.

4.4.3. RQ 4.3 : How much empirical evidence exists for each category of techniques?

In the paper by Doğan et al. (2014) there were many studies that supported dynamic web applications, but also many that only worked on static web applications. Dynamic web applications are systems that offer different content depending on the context, for example they show different content for different users. Static websites on the other hand, will always display exactly the same content, regardless of the situation, the page is retrieved in. The situation has changed in the studies, we extracted. All of our studies support both static and dynamic web applications.

The same result occurs for synchronous versus asynchronous web applications. In synchronous systems, all content for a page is delivered through one request. In asynchronous systems, however, additional content can be loaded using Ajax requests. While some studies only worked on synchronous web applications in the original study by Doğan et al. (2014), all of our extracted studies are now able to work with both synchronous and asynchronous web applications.

Fig. 17 shows the papers that used and evaluated techniques only on client side, server side or on both sides. This result is a little bit different from Doğan et al. (2014). Before we had a high number of studies that worked only on the server side, this has changed in our research. We can assume, that this result is connected to the results on dynamic webpages and the use of asynchronous techniques. Client and server are more connected and the client side gets bigger roles in web applications through the use of Ajax.

4.5. RQ 5 — Data and Validation applications

To validate new methods and techniques, it is important to test them against datasets that are comparable. In the case of web application testing, these data sets are usually test applications that present different challenges to the testing methods. We extracted the most used validation applications that were used in the extracted papers (presented in Table 9). Fig. 18 shows how often these projects were used in our extracted papers.

It should be noted that four of the seven most used validation applications have not been updated since 2017 or are even older. The advantage of this fact is, that we can assume a certain maturity and stability for these test applications. In addition, different methodologies tested on the same versions of test applications become more comparable. The disadvantage however is that new technologies cannot be tested with these rather old projects. This is also reflected by the fact that all the most used applications are PHP projects, besides on Java project. PHP was over a long period the preferred language to implement smaller projects easily. Different languages and frameworks are now more usual than earlier for web application implementation. They offer different opportunities, but also pose different challenges. It would be preferable to see a little bit more variety in implementation languages and age of test projects.

A short overview of the seven most used projects is provided. Claroline (Softaculous, 2024) is an Open Source eLearning platform, developed in PHP/MySQL. It has about 300000 lines of code. AddressBook (PHPAddressBook, 2024) is an Open Source PHP/MySQL application with a mature code base and about 20000 lines of code. Recent developments have stopped with the last update in 2017. This is in fact an advantage in the context of comparing methods against each other, since one can assume to test against the same code base. Mantis Bug Tracker is an open source bug tracking system written in PHP. It is still in active development and a good candidate for testing. Petclinic is the only Java application in this list of the most used applications. It is build on modern frameworks both on server and client side. Collabtive (Collabtive, 2024) a collaboration software to manage projects, tasks, files, time tracking, instant messages — 60000 lines of code. WordPress is a PHP framework to host web applications, but is a web application in itself to administrate pages and other content. The size and complexity depends on plugins and size of hosted application. Schoolmate (SchoolMate, 2024) is an Open Source PHP/MySQL application written with CodeIgniter with about 8000 lines of code. There are different roles for teachers, students and parents and different views, functions and actions per role.

The most used applications for validation are PHP applications. This has some implications for testing. Certain server side problems do not occur or occur more often when validation is done only on certain frameworks or server side languages. Further modern tools or frameworks that are available for other server side languages remain untested. Another problem is, that there is no set of explicitly documented errors with solutions to test on. The only way to test the quality of a testing framework, is to introduce errors, *e.g.* by mutation testing. With a set of applications and described errors, test frameworks would be easier to compare.

Soto-Sánchez et al. (2022) compiled such a set of web applications written in Java which use modern technology and can compliment testing with the established test applications. They have implemented a social network, an online course and a library web application with different versions and regression bugs to test on. Using this set of applications for method validation would benefit future research, since they offer a more modern test application framework.

Another form of strong validation for the methods would be to test them on applications used in the industry. As shown in RQ 3.3 this is not done in most of the analyzed studies and shows an important deficiency of research papers on this topic. Both directions are important, open high quality validation applications to make the comparison of techniques easier and testing on industrial applications to show the applicability.

Table 9
Most used test application for validation. The number for lines of code is an approximate number and depends on the version used in the respective paper.

Application	Updated	Access	Language	LOC
Claroline	2017	https://www.softaculous.com/apps/educational/Claroline	PHP	300 000
AddressBook	2017	https://sourceforge.net/projects/php-addressbook	PHP	20 000
Mantisbt	2024	https://mantisbt.org	PHP	200 000
Petclinic	2024	https://github.com/spring-projects/spring-petclinic	Java	12 000
Collabtive	2017	https://sourceforge.net/projects/collabtive/	PHP	60 000
WordPress	2024	https://wordpress.org/	PHP	800 000
Schoolmate	2013	https://sourceforge.net/projects/schoolmate/	PHP	8000

In addition to the use of standard applications with an industrial level size for testing, we recommend that authors should provide the codebase in their replication package, with a reference to the specific commit-Id in the repository. The codebase is important, because the commit-Id might disappear from the repository, or even the repository itself might be deleted. A good approach to avoid such issues (although the codebase should still be provided in the repository package) is to rely on the Software Heritage project to archive the repository (Software Heritage Project, 2024).

> Many applications that are used for validation are outdated. There are no standard applications used for tests with pre-defined errors to test on.

4.6. Threats to validity

There are different threats to validity that have to be addressed.

4.6.1. Internal threats

Firstly, the conducted extraction process is susceptible to the possibility of overlooking potentially important journal papers. Both the selection of sources for extraction and the execution itself carry risks. In terms of source selection, there is a chance that some papers were not included in the source library at all, or they may be ranked lower compared to other sources.

The search parameters have significant impacts on the results, and slight changes can lead to the omission of papers.

The third point during the extraction process is that with a large number of results (several thousand) from some sources, a cutoff had to be made, and only papers in the upper range of the result list were considered. We can assume that the sources sorted the results by relevance to the search parameters. This means papers later in the results become less relevant. This aligns with our experience in reviewing the results. However, there is a possibility that a paper ranked lower down could have been relevant but was not taken into consideration by us.

Regarding the filtering of papers in terms of quality, there are risks to the validity as well. The assessment of rigor and industrial relevance was made based on subjective criteria, which, although well-formulated, may not have been definitively determinable. Important papers that provide crucial new techniques but were poorly experimentally validated could have been overlooked as a result. Since one author performed the quality assessment while the other author checked the assessment, errors in the filter process were possible.

Other risks exist where papers had to be classified into categories. The classifications for test model, failure model or problem as an example are subjective for some of the papers. Even the description of the given problem classes is already a subjective decision we had to make. Since one author did the classification and the other author checked the categories and decisions ex post, this opens the possibility for assessment errors.

In cases where we did provide a fixed value (*e.g.* rigor or industrial relevance) some of the papers gave not sufficient information, and we had to infer a value from the context (*e.g.* guessing the size of the validation application, when LOC was not provided).

To address these problems, a reproduction package can be obtained (Reproduction Package, 2024) with all extracted papers and their classifications and evaluations to enable a replication of the results.

4.6.2. External and conclusion threats

Since we only had 72 papers, our conclusions are based on these extracted papers. Tendencies or conclusions we have drawn from the data are therefore limited and cannot be generalized to other research in the field. We tried to extract papers that both had most relevance and a high level of quality, as described by our exclusion criteria.

We have only analyzed papers in the domain of web application testing in this systematic literature study. The data, results and conclusions are therefore only applicable and valid in the field of web application testing. Since we followed a systematic procedure, we think our study is repeatable to some extent, with the exception of the data extraction process, since it depends on the external data sources. An extraction at later times may lead to different results.

5. Conclusion

In this paper we conducted a systematic literature review as described by Kitchenham and Charters (2007). From initially over 6000 papers, that have been extracted from different sources, 72 papers were chosen after a filter process and analyzed for different research questions.

We identified the important journals and researchers in the field of web application testing. The number of published papers did not decrease over the observed time. We found groups of researchers that work together and were responsible for about a fifth of our extracted papers. We analyzed further the journals and found that the top three journals published almost 50 percent of our journal papers. We can conclude that there are few main contributors and journals that are responsible for the main work in this field.

We looked further into problems, testing techniques and tools. The main problems have been identified and strategies to answer the problems have been described. The main questions center around the generation of test models, the efficiency of tests and how to deal with test regressions. Further, a description of test models and fault models was given. Most studies use DOM based models for their test suites. Regarding the developed tools, we found, that more tools are open access than earlier. This is a positive trend in development.

Next, the design and report of the empirical studies of the extracted papers were examined. We extracted information on metrics for cost and effectiveness. Most of the papers use time efficiency metrics to measure cost and some kind of coverage metric for efficiency. We also

viewed at the threats for validity. Compared with the results in an earlier review, the number of papers that mention threats for validity has increased. This is also a positive trend for quality. Rigor and relevance have also been examined. We found that most papers have only little industrial relevance since validation of techniques was not done in an industrial setting, but often done only on small validation application examples.

We looked also on empirical evidence. Regarding scalability, only three papers were found, that mentioned scalability. This is definitely a gap in research. We also investigated if results for different techniques have been compared with other techniques. However, we found, that only 40% of our papers had comparisons of techniques. Further, all techniques are applicable for dynamic and asynchronous web applications. This was not the case ten years ago. We see also a shift in techniques from the server side to the client side, or to techniques that work on both as a whole.

We found a research gap in the use of validation applications. Most of the extracted papers use applications to validate data that are not updated recently, and are no longer representing state of the art of web applications. Modern approaches and test applications to validate techniques are needed.

For developers and testers in the industry, there are many tools and methods that look promising and should be used in real application testing. It would be interesting to find more case studies from the industry about their usage and application.

As implications for researchers, they should aim for higher industrial relevance by validating their methods on industrial-level validation applications or at least on modern test applications with predefined errors and solutions. Methods should be compared with similar methods to place the results in context. Threats to validity should be discussed to find possible flaws, and quality and efficiency metrics should be chosen with care to compare results with other methods.

For future research, besides further development of already existing methods and tools, the gap between industry and research should be analyzed more deeply to find solutions. It would be interesting to test techniques of the different research areas on bigger applications and compare their results. Another observation that would be interesting to research is that we only found few papers using modern machine learning techniques for testing. Web application testing as a way to ensure the quality of these systems is a research field with many problems still to solve, while web applications and software architecture as a subject of research are changing over time.

CRediT authorship contribution statement

Sebastian Balsam: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Data curation, Conceptualization. **Deepti Mishra:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table 10
Efficiency metrics for the extracted papers.

Efficiency metric	Metric name	Paper
Coverage	Number explored pages	Bures et al. (2018)
	test coverage	Sprenkle et al. (2013), Thummalapenta et al. (2013b)
	robustness(percentage of located elements)	Leotta et al. (2016)
	functional coverage	Qi et al. (2019)
	code coverage	Mattiello and Endo (2022), Wang et al. (2016), Nguyen et al. (2019), Sherin et al. (2021), Qi et al. (2017), Andrews et al. (2019)
	Exploration tree depth and width	Leveau et al. (2020)
	output coverage	Nguyen et al. (2019)
	Navigational coverage	Sherin et al. (2023),Biagiola et al. (2019b)
	Html coverage	Sherin et al. (2021)
	identified endpoints	Leithner and Simos (2021)
	found page objects and locators	Leotta et al. (2018)
	DOM coverage	Sherin et al. (2021), Imtiaz et al. (2021), Zou et al. (2014), Li et al. (2014a), Sung et al. (2016), Fard et al. (2015)
	V-DOM coverage	Sherin et al. (2021)
	statement coverage	Sherin et al. (2021)
	branch coverage	Zheng et al. (2021) Biagiola et al. (2019b)
	completeness	Qi et al. (2017)
	coverage by tests	Leotta et al. (2023)
	Number of distinct DOM states	Sherin et al. (2023)
	DOM diversity	Sherin et al. (2023)
	number of generated tests	Stocco et al. (2017), Yousaf et al. (2019)
	test size	Soto-Sánchez et al. (2022)
	number of extracted features	Long et al. (2022)
	Navigational diversity	Sherin et al. (2023)
	fitness functions found	Abadeh (2021)
	number of test case pairs	Zhou et al. (2016)
Fault detection	Average Percentage of Fault Detection (APFD)	Mohd-Shafie et al. (2020)
	Percentage identified	Ocariza et al. (2016),Corazza et al. (2021)
	number of mutant	Mirshokraie et al. (2015), Leotta et al. (2023)
	percentage of broken locator	Leotta et al. (2021)

(*continued on next page*)

Table 10 (*continued*).

Efficiency metric	Metric name	Paper
	Fault detection	Habibi and Mirian-Hosseinabadi (2015), Imtiaz et al. (2021), Habibi and Mirian-Hosseinabadi (2015),Thummalapenta et al. (2013a), Maezawa et al. (2013), Choudhary et al. (2013), Fard et al. (2014), Ocariza et al. (2017), Kwon et al. (2018), Kwon et al. (2018), Stocco et al. (2018)
	mutation score	Habibi and Mirian-Hosseinabadi (2015)
	F-score, Recall, Precision	Nguyen et al. (2021), Yandrapally et al. (2020), Bajaj et al. (2015)
	number of defects	Boukhris et al. (2017)
	reduction of I18n errors	Mahajan et al. (2021)
	Error states	Sherin et al. (2023)
	Fault Severity	Mirshokraie et al. (2015)
	number of repaired tests	Imtiaz et al. (2021) Long et al. (2020), Hammoudi et al. (2016), Qi et al. (2023)
	test repair quality	Imtiaz et al. (2021)
	keyboard failures, traps	Chiou et al. (2023), Chiou et al. (2021)
	number of classifications	Zhang and Wang (2017)
	Correctness	Schur et al. (2013), Mirzaaghaei and Mesbah (2014)
Time efficiency	time efficiency	Eda and Do (2019), Biagiola et al. (2019a),Guarnieri et al. (2017)
	execution time	Choudhary et al. (2014)
	smallest iteration for cost	Khanna et al. (2019)
	test path reduction	Do and Hossain (2014)
	reuse of tests	Eda and Do (2019)
	number of calls	Yandrapally et al. (2015)
Comfort	comfort in writing tests	Bünder and Kuchen (2019)
Others	found inputs	Mariani et al. (2014), Clerissi et al. (2020)
	output variability	Nguyen et al. (2014)

References

Abadeh, Maryam Nooraei, 2021. Genetic-based web regression testing: An ontology-based multi-objective evolutionary framework to auto-regression testing of web applications. Serv. Orient. Comput. Appl. 15 (1), 55–74.

Adamsen, Christoffer Quist, Møller, Anders, Alimadadi, Saba, Tip, Frank, 2018. Practical AJAX race detection for JavaScript web applications. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, Lake Buena Vista FL USA, pp. 38–48.

Alenzi, Abdullah, Alhumud, Waleed, Bryce, Renee, Alshammari, Nasser, 2022. A survey of software testing tools in the web development domain. Consortium Comput. Sci. Colleges 38 (2), 63–73.

Alshahwan, Nadia, Harman, Mark, Marginean, Alexandru, 2023. Software testing research challenges : An industrial perspective. In: 2023 IEEE Conference on Software Testing , Verification and Validation. ICST, IEEE, Dublin, Ireland, pp. 1–10.

Andrews, Anneliese, Alhaddad, Ahmed, Boukhris, Salah, 2019. Black-box model-based regression testing of fail-safe behavior in web applications. J. Syst. Softw. 149, 318–339.

Anwar, Zeeshan, Afzal, Hammad, Bibi, Nazia, Abbas, Haider, Mohsin, Athar, Arif, Omar, 2019. A hybrid-adaptive neuro-fuzzy inference system for multi-objective regression test suites optimization. Neural Comput. Appl. 31 (11), 7287–7301.

Artzi, Shay, Dolby, Julian, Jensen, Simon Holm, Møller, Anders, Tip, Frank, 2011. A framework for automated testing of javascript web applications. In: Proceedings of the 33rd International Conference on Software Engineering. ACM, Waikiki, Honolulu HI USA, pp. 571–580.

Aydos, Murat, Aldan, Çiğdem, Coşkun, Evren, Soydan, Alperen, 2022. Security testing of web applications: A systematic mapping of the literature. J. King Saud Univ. -Comput. Inf. Sci. 34 (9), 6775–6792.

Bajaj, Kartik, Pattabiraman, Karthik, Mesbah, Ali, 2015. Synthesizing web element locators (T). In: 2015 30th IEEE /ACM International Conference on Automated Software Engineering. ASE, IEEE, Lincoln, NE, USA, pp. 331–341.

Benedikt, Michael, Freire, Juliana, Godefroid, Patrice, 2002. VeriWeb : Automatically testing dynamic web sites. In: Proc. the 11th Int. Conf. World Wide Web. pp. 654–668.

Biagiola, Matteo, Stocco, Andrea, Mesbah, Ali, Ricca, Filippo, Tonella, Paolo, 2019a. Web test dependency detection. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, Tallinn Estonia, pp. 154–164.

Biagiola, Matteo, Stocco, Andrea, Ricca, Filippo, Tonella, Paolo, 2019b. Diversity-based web test generation. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, Tallinn Estonia, pp. 142–153.

Biørn-Hansen, Andreas, Majchrzak, Tim A., Grønli, Tor-Morten, 2017. Progressive web apps : The possible web-native unifier for mobile development. In: Proceedings of the 13th International Conference on Web Information Systems and Technologies. SCITEPRESS - Science and Technology Publications, Porto, Portugal, pp. 344–351.

Boukhris, Salah, Andrews, Anneliese, Alhaddad, Ahmed, Dewri, Rinku, 2017. A case study of black box fail-safe testing in web applications. J. Syst. Softw. 131, 146–167.

Brereton, Pearl, Kitchenham, Barbara A., Budgen, David, Turner, Mark, Khalil, Mohamed, 2007. Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. 80 (4), 571–583.

Bünder, Hendrik, Kuchen, Herbert, 2019. Towards behavior-driven graphical user interface testing. ACM SIGAPP Appl. Comput. Rev. 19 (2), 5–17.

Bures, Miroslav, Frajtak, Karel, Ahmed, Bestoun S., 2018. Tapir: Automation support of exploratory testing using model reconstruction of the system under test. IEEE Trans. Reliab. 67 (2), 557–580.

Chiou, Paul T., Alotaibi, Ali S., Halfond, William G.J., 2023. Detecting dialog-related keyboard navigation failures in web applications. In: 2023 IEEE /ACM 45th International Conference on Software Engineering. ICSE, IEEE, Melbourne, Australia, pp. 1368–1380.

Chiou, Paul T., Alotaibi, Ali S., William G.J. Halfond, 2021. Detecting and localizing keyboard accessibility failures in web applications. In: Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, Athens Greece, pp. 855–867.

Choudhary, Suryakant, Dincturk, Emre, Mirtaheri, Seyed, 2014, Model-Based Rich Internet Applications Crawling : Menu and Probability Models.

Choudhary, Shauvik Roy, Prasad, Mukul R., Orso, Alessandro, 2013. Orso X-PERT : Accurate identification of cross-browser issues in web applications. In: 2013 35th International Conference on Software Engineering. ICSE, IEEE, San Francisco, CA, USA, pp. 702–711.

Choudhary, Shauvik Roy, Zhao, Dan, Versee, Husayn, Orso, Alessandro, 2011. WATER : Web application test repair. In: Proceedings of the First International Workshop on End-To-End Test Script Engineering. ACM, Toronto Ontario Canada, pp. 24–29.

Clerissi, Diego, Denaro, Giovanni, Mobilio, Marco, Mariani, Leonardo, 2020. Plug the database & play with automatic testing: Improving system testing by exploiting persistent data. In: Proceedings of the 35th IEEE / ACM International Conference on Automated Software Engineering. ACM, Virtual Event Australia, pp. 66–77.

Collabtive, https://sourceforge.net/projects/collabtive/.

Coppola, Riccardo, Alégroth, Emil, 2022. A taxonomy of metrics for GUI-based testing research: A systematic literature review. Inf. Softw. Technol. 152, 107062.

Corazza, Anna, Martino, Sergio Di, Peron, Adriano, Starace, Luigi Libero Lucio, 2021. Web application testing : Using tree kernels to detect near-duplicate states in automated model inference. In: Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM, ACM, Bari Italy, pp. 1–6.

Cruzes, D.S., Dyba, T., 2011. Recommended steps for thematic synthesis in software engineering. In: 2011 International Symposium on Empirical Software Engineering and Measurement. IEEE, Banff, AB, pp. 275–284.
Dadkhah, Mahboubeh, 2020. A systematic literature review on semantic web enabled software testing.
Dalisay, Ferdinand V., 2018. Issues and challenges of mobile web applications : A systematic literature review. 3, (1).
Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6 (2), 182–197.
Do, Hyunsook, Hossain, Md., 2014. An efficient regression testing approach for PHP web applications: A controlled experiment. Softw. Test. Verif. Reliab. 24 (5), 367–385.
Doğan, Serdar, Betin-Can, Aysu, Garousi, Vahid, 2014. Web application testing: A systematic literature review. J. Syst. Softw. 91, 174–201.
Eda, Ravi, Do, Hyunsook, 2019. An efficient regression testing approach for PHP web applications using test selection and reusable constraints. Softw. Qual. J. 27 (4), 1383–1417.
El-Kassas, Wafaa S., Abdullah, Bassem A., Yousef, Ahmed H., Wahba, Ayman M., 2017. Taxonomy of cross-platform mobile applications development approaches. Ain Shams Eng. J. 8 (2), 163–190.
Fard, Amin Milani, Mesbah, Ali, 2013. Feedback-directed exploration of web applications to derive test models. In: 2013 IEEE 24th International Symposium on Software Reliability Engineering. ISSRE, IEEE, Pasadena, CA, pp. 278–287.
Fard, Amin Milani, Mesbah, Ali, Wohlstadter, Eric, 2015. Generating fixtures for JavaScript unit testing (T). In: 2015 30th IEEE /ACM International Conference on Automated Software Engineering. ASE, IEEE, Lincoln, NE, USA, pp. 190–200.
Fard, Amin.Milani, Mirzaaghaei, Mehdi, Mesbah, Ali, 2014. Leveraging existing tests in automated test generation for web applications. In: Proceedings of the 29th ACM / IEEE International Conference on Automated Software Engineering. ACM, Vasteras Sweden, pp. 67–78.
Garousi, Vahid, Felderer, Michael, 2017. Worlds apart : Industrial and Academic Focus Areas in software testing. IEEE Softw. 34 (5), 38–45.
Garousi, Vahid, Felderer, Michael, Kuhrmann, Marco, Herkiloğlu, Kadir, Eldh, Sigrid, 2020. Exploring the industry's challenges in software testing: An empirical study. J. Softw. Evol. Process 32 (8), e2251.
Garousi, Vahid, Mesbah, Ali, Betin-Can, Aysu, Mirshokraie, Shabnam, 2013. A systematic mapping study of web application testing. Inf. Softw. Technol. 55 (8), 1374–1396.
Garrett, Jesse James, 2005. Ajax: A new approach to web applications.
Guarnieri, Marco, Tsankov, Petar, Buchs, Tristan, Dashti, Mohammad Torabi, Basin, David, 2017. Test execution checkpointing for web applications. In: Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis. ACM, Santa Barbara CA USA, pp. 203–214.
Habibi, Elahe, Mirian-Hosseinabadi, Seyed-Hassan, 2015. Event-driven web application testing based on model-based mutation testing. Inf. Softw. Technol. 67, 159–179.
Hammoudi, Mouna, Rothermel, Gregg, Stocco, Andrea, 2016. WATERFALL : An incremental approach for repairing record-replay tests of web applications. In: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, Seattle WA USA, pp. 751–762.
Hoffswell, Jane, Li, Wilmot, Liu, Zhicheng, 2020. Techniques for flexible responsive visualization design. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. ACM, Honolulu HI USA, pp. 1–13.
Imtiaz, Javaria, Iqbal, Muhammad Zohaib, Khan, Muhammad Uzair, 2021. An automated model-based approach to repair test suites of evolving web applications. J. Syst. Softw. 171, 110841.
Ivarsson, Martin, Gorschek, Tony, 2011. Empir. Softw. Eng. 16 (3), 365–395.
Jameel, Asad, Shahzad, Khurram, Zafar, Afia, Ahmed, Usman, Hussain, Syed Jawad, Sajid, Ahthasham, 2018. The users experience quality of responsive web design on multiple devices. In: Proceedings of the 2nd International Conference on Future Networks and Distributed Systems. ACM, Amman Jordan, pp. 1–6.
JBehave, What is JBehave?, https://jbehave.org/.
Jensen, Casper S., Møller, Anders, Su, Zhendong, 2013. Server interface descriptions for automated testing of JavaScript web applications. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. ACM, Saint Petersburg Russia, pp. 510–520.
Khanna, Munish, Chaudhary, Achint, Toofani, Abhishek, Pawar, Anil, 2019. Performance comparison of multi-objective algorithms for test case prioritization during web application testing. Arab. J. Sci. Eng. 44 (11), 9599–9625.
Kitchenham, Barbara Ann, Charters, Stuart, 2007. Charters Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report.
Kitchenham, B.A., Dyba, T., Jorgensen, M., 2004. Evidence-based software engineering. In: Proceedings. 26th International Conference on Software Engineering. IEEE Comput. Soc, Edinburgh, UK, pp. 273–281.
Kong, Pingfan, Li, Li, Gao, Jun, Liu, Kui, Bissyande, Tegawende F., Klein, Jacques, 2019. Automated testing of android apps : A systematic literature review. IEEE Trans. Reliab. 68 (1), 45–66.
Konsaard, Patipat, Ramingwong, Lachana, 2015. Total coverage based regression test case prioritization using genetic algorithm. In: 2015 12th International Conference on Electrical Engineering /Electronics, Computer, Telecommunications and Information Technology. ECTI-CON, IEEE, Hua Hin, Cha-am, Thailand, pp. 1–6.
Kwon, Jung-Hyun, Ko, In-Young, Rothermel, Gregg, 2018. Prioritizing browser environments for web application test execution. In: Proceedings of the 40th International Conference on Software Engineering. ACM, Gothenburg Sweden, pp. 468–479.
Leithner, Manuel, Simos, Dimitris E., 2021. CHIEv : Concurrent hybrid analysis for crawling and modeling of web applications. ACM SIGAPP Appl. Comput. Rev. 21 (1), 5–23.
Leotta, Maurizio, Paparella, Davide, Ricca, Filippo, 2023. Mutta: A novel tool for E2E web mutation testing. Softw. Qual. J..
Leotta, Maurizio, Ricca, Filippo, Tonella, Paolo, 2021. Sidereal: Statistical adaptive generation of robust locators for web testing. Softw. Test. Verif. Reliab. 31 (3).
Leotta, Maurizio, Stocco, Andrea, Ricca, Filippo, Tonella, Paolo, 2016. ROBULA+ : An algorithm for generating robust xpath locators for web testing: ROBULA +: An algorithm for generating robust xpath locators. J. Softw. Evol. Process 28 (3), 177–204.
Leotta, Maurizio, Stocco, Andrea, Ricca, Filippo, Tonella, Paolo, 2018. P ESTO : Automated migration of DOM-based web tests towards the visual approach: P ESTO : Automated migration of DOM-based web tests towards the visual approach. Softw. Test. Verif. Reliab. 28 (4), e1665.
Leveau, Julien, Blanc, Xavier, Reveillere, Laurent, Falleri, Jean-Remy, Rouvoy, Romain, 2020. Fostering the diversity of exploratory testing in web applications. In: 2020 IEEE 13th International Conference on Software Testing , Validation and Verification. ICST, IEEE, Porto, Portugal, pp. 164–174.
Li, Guodong, Andreasen, Esben, Ghosh, Indradeep, 2014a. SymJS : Automatic symbolic testing of JavaScript web applications. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, Hong Kong China, pp. 449–459.
Li, Yuan-Fang, Das, Paramjit K., Dowe, David L., 2014b. Two decades of web application testing—A survey of recent advances. Inf. Syst. 43, 20–54.
Long, Yong-Hao, Chen, Yan-Cheng, Chen, Xiang-Ping, Shi, Xiao-Hong, Zhou, Fan, 2022. Test-driven feature extraction of web components. J. Comput. Sci. Tech. 37 (2), 389–404.
Long, Zhenyue, Wu, Guoquan, Chen, Xiaojiang, Chen, Wei, Wei, Jun, 2020. WebRR : Self-replay enhanced robust record/replay for web application testing. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, Virtual Event USA, pp. 1498–1508.
Madhurima, Chauhan, Anuj Kumar, Dhir, Saru, Madhulika, 2015. Difficulties and challenges faced in testing AJAX applications. In: 2015 International Conference on Soft Computing Techniques and Implementations. ICSCTI, IEEE, Faridabad, India, pp. 113–115.
Maezawa, Yuta, Washizaki, Hironori, Tanabe, Yoshinori, Honiden, Shinichi, 2013. Automated verification of pattern-based interaction invariants in Ajax applications. In: 2013 28th IEEE /ACM International Conference on Automated Software Engineering. ASE, IEEE, Silicon Valley, CA, USA, pp. 158–168.
Mahajan, Sonal, Alameer, Abdulmajeed, McMinn, Phil, Halfond, William G.J., 2018. Automated repair of internationalization presentation failures in web pages using style similarity clustering and search-based techniques. In: 2018 IEEE 11th International Conference on Software Testing, Verification and Validation. ICST, IEEE, Vasteras, pp. 215–226.
Mahajan, Sonal, Alameer, Abdulmajeed, McMinn, Phil, Halfond, William G.J., 2021. Effective automated repair of internationalization presentation failures in web applications using style similarity clustering and search-based techniques. Softw. Test. Verif. Reliab. 31 (1–2).
Mariani, Leonardo, Pezzè, Mauro, Riganelli, Oliviero, Santoro, Mauro, 2014. Link: Exploiting the web of data to generate test inputs. In: Proceedings of the 2014 International Symposium on Software Testing and Analysis. ACM, San Jose CA USA, pp. 373–384.
Mattiello, Guilherme Ricken, Endo, André Takeshi, 2022. Model-based testing leveraged for automated web tests. Softw. Qual. J. 30 (3), 621–649.
Mesbah, Ali, Deursen, Arie Van, Lenselink, Stefan, 2012. Crawling ajax-based web applications through dynamic analysis of user interface state changes. ACM Trans. Web 6 (1), 1–30.
Mirshokraie, Shabnam, Mesbah, Ali, Pattabiraman, Karthik, 2015. Guided mutation testing for JavaScript web applications. IEEE Trans. Softw. Eng. 41 (5), 429–444.
Mirzaaghaei, Mehdi, Mesbah, Ali, 2014. DOM-based test adequacy criteria for web applications. In: Proceedings of the 2014 International Symposium on Software Testing and Analysis. ACM, San Jose CA USA, pp. 71–81.
Mohd-Shafie, Muhammad Luqman, Wan-Kadir, Wan Mohd Nasir, Khatib-syarbini, Muhammad, Isa, Mohd Adham, 2020. Model-based test case prioritization using selective and even-spread count-based methods with scrutinized ordering criterion. Plos One 15 (2), e0229312.
Molina-Ríos, Jimmy, Pedreira-Souto, Nieves, 2020. Comparison of development methodologies in web applications. Inf. Softw. Technol. 119, 106238.
Montoto, Paula, Pan, Alberto, Raposo, Juan, Bellas, Fernando, López, Javier, 2011. Automated browsing in AJAX websites. Data Knowl. Eng. 70 (3), 269–283.
Nass, Michel, Alégroth, Emil, Feldt, Robert, Leotta, Maurizio, Ricca, Filippo, 2023. Similarity-based web element localization for robust test automation. ACM Trans. Softw. Eng. Methodol. 32 (3), 1–30.

Nguyen, Hung.Viet, Kästner, Christian, Nguyen, Tien N., 2014. Exploring variability-aware execution for testing plugin-based web applications. In: Proceedings of the 36th International Conference on Software Engineering. ACM, Hyderabad India, pp. 907–918.

Nguyen, Hung Viet, Phan, Hung Dang, Kästner, Christian, Nguyen, Tien N., 2019. Exploring output-based coverage for testing PHP web applications. Autom. Softw. Eng. 26 (1), 59–85.

Nguyen, Vu, To, Thanh, Diep, Gia-Han, 2021. Generating and selecting resilient and maintainable locators for web automated testing. Softw. Test. Verif. Reliab. 31 (3).

Ocariza, Frolin S., Li, Guanpeng, Pattabiraman, Karthik, Mesbah, Ali, 2016. Automatic fault localization for client-side JavaScript. Softw. Test. Verif. Reliab. 26 (1), 69–88.

Ocariza, Frolin S., Pattabiraman, Karthik, Mesbah, Ali, 2017. Detecting unknown inconsistencies in web applications. In: 2017 32nd IEEE /ACM International Conference on Automated Software Engineering. ASE, IEEE, Urbana, IL, pp. 566–577.

Palomino, Fryda, Paz, Freddy, Moquillaza, Arturo, 2021. Web analytics for user experience : A systematic literature review. In: Soares, Marcelo M., Rosenzweig, Elizabeth, Marcus, Aaron (Eds.), In: Design, User Experience, and Usability : UX Research and Design, vol. 12779, Springer International Publishing, Cham, pp. 312–326.

Panda, Namita, Mohapatra, Durga Prasad, 2021. Test scenario prioritization from user requirements for web-based software. Int. J. Syst. Assur. Eng. Manag. 12 (3), 361–376.

PHPAddressBook, https://sourceforge.net/projects/php-addressbook/.

Prokhorenko, Victor, Choo, Kim-Kwang Raymond, Ashman, Helen, 2016. Web application protection techniques: A taxonomy. J. Netw. Comput. Appl. 60, 95–112.

Qi, Xiao-Fang, Hua, Yun-Long, Wang, Peng, Wang, Zi-Yuan, 2019. Leveraging keyword-guided exploration to build test models for web applications. Inf. Softw. Technol. 111, 110–119.

Qi, Xiaofang, Qian, Xiang, Li, Yanhui, 2023. Semantic test repair for web applications. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, San Francisco CA USA, pp. 1190–1202.

Qi, Xiao-Fang, Wang, Zi-Yuan, Mao, Jun-Qiang, Wang, Peng, 2017. Automated testing of web applications using combinatorial strategies. J. Comput. Sci. Tech. 32 (1), 199–210.

Reproduction Package, https://github.com/sbalsam/JOSAS-ReproPackage2024.

Ricca, Filippo, Leotta, Maurizio, Stocco, Andrea, 2019. Three open problems in the context of E2E web testing and a vision : NEONATE. In: Advances in Computers, vol. 113, Elsevier, pp. 89–133.

Romano, Alan, Song, Zihe, Grandhi, Sampath, Yang, Wei, Wang, Weihang, 2021. An empirical analysis of UI-based flaky tests. In: 2021 IEEE /ACM 43rd International Conference on Software Engineering. ICSE, IEEE, Madrid, ES, pp. 1585–1597.

SchoolMate, https://sourceforge.net/projects/schoolmate/.

Schur, Matthias, Roth, Andreas, Zeller, Andreas, 2013. Mining behavior models from enterprise web applications. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. ACM, Saint Petersburg Russia, pp. 422–432.

Sherin, Salman, Iqbal, Muhammad Zohaib, Khan, Muhammad Uzair, Jilani, Atif Aftab, 2021. Comparing coverage criteria for dynamic web application: An empirical evaluation. Comput. Stand. Interfaces 73, 103467.

Sherin, Salman, Muqeet, Asmar, Khan, Muhammad Uzair, Iqbal, Muhammad Zohaib, 2023. QExplore : An exploration strategy for dynamic web applications using guided search. J. Syst. Softw. 195, 111512.

Softaculous, https://www.softaculous.com/apps/educational/Claroline.

Software Heritage Project, https://www.softwareheritage.org/.

Soto-Sánchez, Óscar, Maes-Bermejo, Michel, Gallego, Micael, Gortázar, Francisco, 2022. A dataset of regressions in web applications detected by end-to-end tests. Softw. Qual. J. 30 (2), 425–454.

Sprenkle, Sara E., Pollock, Lori L., Simko, Lucy M., 2013. Configuring effective navigation models and abstract test cases for web applications by analysing user behaviour. Softw. Test. Verif. Reliab. 23 (6), 439–464.

Stocco, Andrea, Leotta, Maurizio, Ricca, Filippo, Tonella, Paolo, 2017. APOGEN : Automatic page object generator for web testing. Softw. Qual. J. 25 (3), 1007–1039.

Stocco, Andrea, Yandrapally, Rahulkrishna, Mesbah, Ali, 2018. Visual web test repair. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, Lake Buena Vista FL USA, pp. 503–514.

Sung, Chungha, Kusano, Markus, Sinha, Nishant, Wang, Chao, 2016. Static DOM event dependency analysis for testing web applications. In: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, Seattle WA USA, pp. 447–459.

Thummalapenta, Suresh, Devaki, Pranavadatta, Sinha, Saurabh, Chandra, Satish, Gnanasundaram, Sivagami, Nagaraj, Deepa D., Kumar, Sampath, Kumar, Sathish, 2013a. Efficient and change-resilient test automation: An industrial case study. In: 2013 35th International Conference on Software Engineering. ICSE, IEEE, San Francisco, CA, pp. 1002–1011.

Thummalapenta, Suresh, Vasanta Lakshmi, K., Sinha, Saurabh, Sinha, Nishant, Chandra, Satish, 2013b. Guided test generation for web applications. In: 2013 35th International Conference on Software Engineering. ICSE, IEEE, San Francisco, CA, USA, pp. 162–171.

W3afOpenSource, W3af - Open Source Web Application Security Scanner, http://w3af.org/.

Wang, Wenhua, Sampath, Sreedevi, Lei, Yu, Kacker, Raghu, Kuhn, Richard, Lawrence, James, 2016. Using combinatorial testing to build navigation graphs for dynamic web applications. Softw. Test. Verif. Reliab. 26 (4), 318–346.

Wang, Weiwei, Wu, Shumei, Li, Zheng, Zhao, Ruilian, 2023. Parallel evolutionary test case generation for web applications. Inf. Softw. Technol. 155, 107113.

Wirtz, Bernd W., Schilke, Oliver, Ullrich, Sebastian, 2010. Strategic development of business models. Long Range Plan. 43 (2–3), 272–290.

Yandrapally, Rahulkrishna, Sridhara, Giriprasad, Sinha, Saurabh, 2015. Automated modularization of GUI test cases. In: 2015 IEEE /ACM 37th IEEE International Conference on Software Engineering. IEEE, Florence, Italy, pp. 44–54.

Yandrapally, Rahulkrishna, Stocco, Andrea, Mesbah, Ali, 2020. Near-duplicate detection in web app model inference. In: Proceedings of the ACM /IEEE 42nd International Conference on Software Engineering. ACM, Seoul South Korea, pp. 186–197.

Yousaf, Nazish, Azam, Farooque, Butt, WasiHaider, Anwar, Muhammad Waseem, Rashid, Muhammad, 2019. Automated model-based test case generation for web user interfaces (WUI) from interaction flow modeling language (IFML) models. IEEE Access 7, 67331–67354.

Zalewski, M., Heinen, N., Roschke, S., 0000. Google Code Archive - Long-term storage for Google Code Project Hosting, https://code.google.com/archive/p/skipfish/.

Zhang, Lu, Wang, Chao, 2017. RClassify : Classifying race conditions in web applications via deterministic replay. In: 2017 IEEE /ACM 39th International Conference on Software Engineering. ICSE, IEEE, Buenos Aires, pp. 278–288.

Zheng, Yan, Liu, Yi, Xie, Xiaofei, Liu, Yepang, Ma, Lei, Hao, Jianye, Liu, Yang, 2021. Automatic web testing using curiosity-driven reinforcement learning. In: 2021 IEEE /ACM 43rd International Conference on Software Engineering. ICSE, IEEE, Madrid, ES, pp. 423–435.

Zhou, Zhi Quan, Xiang, Shaowen, Chen, Tsong Yueh, 2016. Metamorphic testing for software quality assessment : A study of search engines. IEEE Trans. Softw. Eng. 42 (3), 264–284.

Zou, Yunxiao, Chen, Zhenyu, Zheng, Yunhui, Zhang, Xiangyu, Gao, Zebao, 2014. Virtual DOM coverage for effective testing of dynamic web applications. In: Proceedings of the 2014 International Symposium on Software Testing and Analysis. ACM, San Jose CA USA, pp. 60–70.

Sebastian Balsam has a degree in Computer Science (Diplom Informatiker) and an M.A. in Philosophy. He is currently working at ETC Electric Time Car AS in Norway and starts his Industrial Ph.D. at the Department of Computer Science (IDI) at the Norwegian University of Science and Technology (NTNU).

Deepti Mishra is a professor in the Department of Computer Science (IDI) at the Norwegian University of Science and Technology (NTNU). She is currently the head of the Intelligent Systems and Analytics (ISA) research group and the Educational Technology Laboratory. She has extensive international experience and has earlier worked at Monash University, Malaysia; Atilim University, Turkey; and various institutions in India. Her research interests include empirical software engineering, artificial intelligence, human-robot interaction, and sustainability.