Journal of Computer Science & Technology, Volume 18, Number 1, April 2018

A Systematic Literature Review in Cross-browser Testing

Leandro N. Sabaren[1], Maximiliano A. Mascheroni[1,2], Cristina L. Greiner[1], Emanuel Irrazábal[1]

[1]*Departamento de Informática. Facultad de Ciencias Exactas y Naturales y Agrimensura. Universidad Nacional del Nordeste. Corrientes, Argentina*
leans177@gmail.com, {mascheroni, cgreiner, eirrazabal}@exa.unne.edu.ar
[2]*Facultad de Informática, Universidad Nacional de La Plata*

Abstract

Many users access web pages from different browsers looking for the same user experience in all of them. However, there are several causes that produce compatibility issues. Those defects affect functionalities and user interface components. In this paper we present a systematic literature review which aims to find and summarize existing techniques, tools and challenges related to cross-browser testing. According to the results, the most used technique is the visual analysis. However, there are still challenges to face. The most important challenge is the identification of dynamic components in the user interface. Cross-browser compatibility topics are getting importance according to an increment in published articles. Nevertheless, there are techniques that are not completely developed yet and do not fully support test automation practices.

Keywords: cross-browser testing, systematic literature review, web application.

1. Introduction

When developing a website, one of the goals is that it has to be visualized by many users worldwide [1, 2]. Due to websites distribution, based on the client-server architecture model [3], users can access any site from many types of web browsers, from different platforms and devices. However, the differences between each browser and the way they interpret the website source code may cause incompatibility defects. One of the developer's tasks is to provide an accepted user experience to every user. Browser compatibility is a website's capability that makes it work correctly in a certain number of web browsers [4]. It is impossible to test a web application in all of the web browsers that exist in the world, and in all operating systems [4]. A website has cross-browser compatibility if it is interpreted in the same way by all of the browsers.

There are test tools in the market, like CrossBrowserTesting[1] or BrowserStacks[2], but they produce screenshots that require visual inspection by a user. In the last years, many researchers have proposed techniques and tools to perform compatibility testing. While the state of the art grows and diversifies, the need to systematically summarize these solutions arises. We propose this study in cross-browser compatibility testing through a systematic literature review, aimed to find: testing techniques, implemented tools and new challenges. This work constitutes the continuation of a previous work published in CACIC 2017 [5], and it includes new content regarding the discussed testing techniques and documented challenges. We also include an updated pool of articles and an additional research line that helps to understand better how the testing tools and techniques were validated.

This paper is divided into sections. The selected methodology is detailed in Section 2. Section 3 lists the selected articles, Section 4 presents the analysis. A list of related works is presented in Section 5. Section 6 contains discussions on findings, research trends and threats to validity. Finally, our conclusions are presented in Section 7.

2. Methodology

The selected methodology is the systematic literature review (SLR), proposed by Kitchenham and Charters [6]. The protocol is shown in Fig. 1.

The goal is to analyze scientific articles related to cross-browser compatibility, focusing in the proposed techniques and tools. The research questions (RQ) will guide the process and the findings will answer the raised questions. We believe that a summary of the state of art will aid

[1]https://crossbrowsertesting.com/
[2]https://www.browserstack.com/

Citation: L. Sabaren, M. Mascheroni, C. Greiner and E. Irrazábal. "*A Systematic Literature Review in Cross-browser Testing*", Journal of Computer Science & Technology, vol. 18, no. 1, pp. 18-27, 2018.

DOI: 10.24215/16666038.18.e03

Received: February 10, 2017 **Revised:** April 04, 2018 **Accepted:** April 06, 2018

works in this line of investigation. The RQ are:

RQ1 - What are the proposed cross-browser compatibility testing techniques?

RQ2 - What are the proposed tools to detect cross-browser incompatibility?

RQ3 - How are the proposed tools validated?

RQ4 - What challenges are presented when implementing the tests approaches?

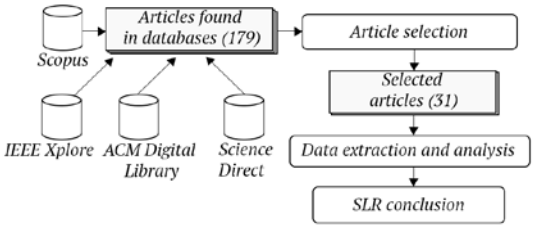

Fig. 1 Diagram of the protocol used for this review.

For the search terms, keywords were extracted from the RQ. Successive searches were conducted, adjusting the parameters to improve the results. The search term was: **(web OR website) AND ("cross-browser" OR "cross browser" OR crossbrowser) AND (test OR testing OR defect OR failure OR issue) AND (technique OR method OR tool).**

Fig. 1 shows the review protocol's steps. After selecting the sources, the search term is used to obtain the articles. Then, articles are selected based on inclusion/exclusion criteria. Once the relevant articles are obtained, pertinent data is extracted and analyzed to answer the RQ. Finally, a summary of them and conclusions are presented. The selected repositories were used in similar studies [7, 8, 9, 10]: Scopus, ACM Digital Library, IEEE Xplore, Science Direct. Relevant studies are selected based on inclusion and exclusion criteria. Every article was examined, taking the title, abstract, keywords, introduction and conclusion sections in order to judge its relevance. The criteria used were:

- Identify article's relevance in the cross-browser compatibility testing domain.
- Evaluate whether the article provides information that addresses the proposed RQ.

3. Selected articles

The first search provided a total of 179 articles. After applying the inclusion/exclusion criteria, 31 articles were selected, as seen in Table 1.

Table 1 Scientific articles selected for this SLR.

ID	Ref.	Title
S1	[11]	A Cross-browser Web Application Testing Tool
S2	[12]	Webdiff: Automated Identification of Cross-browser Issues in Web Applications
S3	[13]	Detecting cross-browser issues in web applications
S4	[14]	Automated cross-browser compatibility testing
S5	[15]	Cross Browser Incompatibility Reasons and Solutions
S6	[16]	CrossCheck: Combining crawling and differencing to better detect cross-browser incompatibilities in web applications
S7	[17]	WebMate: A Tool for Testing Web 2.0 Applications
S8	[18]	Visual testing of Graphical User Interfaces; An exploratory study towards systematic definitions and approaches
S9	[19]	Measuring and Improving Website User Experience using UX Methodologies; A Case Study on Cross Browser Compatibility Heuristic
S10	[20]	X-PERT Accurate identification of cross-browser issues in web applications
S11	[21]	WebMate: Generating test cases for web 2.0
S12	[22]	Browserbite: Accurate cross-browser testing via machine learning over image features
S13	[23]	X-PERT: A web application testing tool for cross-browser inconsistency detection
S14	[24]	Crawl-based analysis of web applications; Prospects and challenges
S15	[25]	Cross Browser Testing Using Automated Test Tools
S16	[26]	Modeling web application for cross-browser compatibility testing
S17	[27]	Finding HTML presentation failures using image comparison techniques
S18	[28]	Adaptive random testing for image comparison in regression web testing
S19	[29]	A crowdsourcing framework for detecting cross-browser issues in web Application
S20	[30]	An oracle based on image comparison for regression testing of web applications
S21	[31]	Detection and Localization of HTML Presentation Failures Using Computer Vision-Based Techniques
S22	[32]	Browserbite: cross-browser testing via image processing
S23	[33]	Cross-Browser Compatibility Testing Based on Model Comparison
S24	[34]	Static Analysis Technique of Cross-Browser Compatibility Detecting
S25	[35]	A Survey on Cross Browser Inconsistencies in Web Application
S26	[36]	X-Check A Novel Cross-browser Testing Service based on Record/Replay
S27	[37]	Using Visual Symptoms for Debugging Presentation Failures in Web Applications
S28	[38]	An Automated Approach for Cross-Browser Inconsistency (XBI) Detection
S29	[39]	Detect cross-browser issues for javascript-based web applications based on record-replay
S30	[40]	Detection of Cross Browser Inconsistency by Comparing Extracted Attributes
S31	[41]	VISOR: A Fast Image Processing Pipeline with Scaling andTranslation Invariance for Test Oracle Automation of Visual OutputSystems

4. Results

4.1. RQ1 – What are the proposed cross-browser compatibility testing techniques?

In total, 29 articles developed testing techniques (93.5%), listed in Table 2. Some of them propose a combination of these. The techniques are:

DOM model analysis. Document Object Model (DOM) is a multi-platform language independent interface to represent HTML, XHTML or XML documents as tree structures [42]. Each tree node represents a webpage's object. This technique compares a pair of models of the same page in different configurations. A configuration is defined by a web browser-operating system pair [22]. In total, 11 studies (35.4%) proposed this technique (S1, S2, S3, S4, S6, S7, S10, S13, S14, S19, S26).

Visual analysis. Consists in comparing screenshots taken from the application. Generally, they are compared in pairs. One image is considered the webpage's correct representation, the second is taken from a different configuration being tested. Methods for image comparison include the use of histograms - color histograms of the screenshots are compared by measuring the χ^2 distance (S13) -, pixel comparison, graphical user interface element's properties comparison and image segmentation in so called, regions of interest. Proposed by 16 studies (51.6%), it is the most selected technique (S1, S2, S6, S10, S13, S3, S8, S12, S17, S21, S18, S20, S22, S26, S27, S31).

Navigation model analysis. Using a web crawler the behavior model of the application is generated, resulting in a finite state machine. These tools explore all the webpage's possible states. This is followed by a comparison of two different models produced. Eight studies (25.8%) proposed this technique (S4, S7, S11, S10, S13, S14, S16, S23).

Record/replay. First, a user performs a series of actions in the application, which are recorded to be replayed in a different configuration. The results are compared to find incompatibilities. Three articles (9.6%) proposed this technique (S19, S26, S29).

Static analysis. A direct analysis of the application's source code is performed, instead of rendering the webpage in a browser. It detects possible conflicting elements bound to the HTML5 standard. The code analysis is managed through regular expressions detection. The researchers first built a database with HTML5 incompatible features linked to the web browsers. This is followed by a detection of HTML5 incompatible features in the application. If an incompatibility is found, a report highlights the code location that generated the issue. The static analysis needs access to the webpage source code to generate tests. One article (3.2%) proposed this technique (S24).

Attribute comparison. It generates graphs using web crawlers, which contain webpage elements' attributes in different configurations. Then, the same element's attributes in different graphs are compared to detect incompatibilities. Two articles (6.4%) proposed this technique (S28, S30).

Heuristic evaluation. It is an inspection method to assess an application's usability, focused on detecting user interface issues. Experts examine the interface, and judge the compliance with predetermined usability principles. The verification lists (defined as heuristic checklists) guide the evaluation process. This is performed across various browsers, resolutions and operating systems. One article (3.2%) proposed this technique (S9).

Table 2 Proposed techniques by selected articles.

Technique	Selected articles	N° of art.
DOM model analysis	(S1, S2, S6, S10, S13, S3, S4, S7, S14, S19, S26)	11
Visual analysis	(S1, S2, S6, S10, S13, S3, S8, S12, S17, S21, S18, S20, S22, S26, S27 S31)	16
Navigation model analysis	(S4, S7, S11, S10, S13, S14, S16, S23)	8
Record/replay	(S19, S26, S29)	3
Static analysis	(S24)	1
Attribute comparison	(S28, S30)	2
Heuristic evaluation	(S9)	1

The tendency in the techniques selection can be seen in Fig. 2. The DOM model analysis was largely preferred the first years. Navigation model analysis had an increase in its use, but it has been overlooked the last two years. Visual analysis was the most selected technique, and its use has been increasing in time. Static analysis and attribute comparison are the newest techniques.

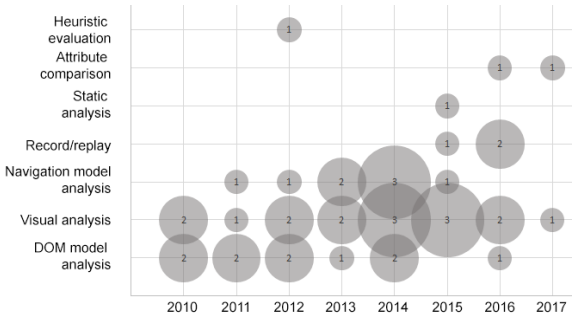

Fig. 2 Number of articles that selected each technique by publication year.

4.2. RQ2 – What are the proposed tools to detect cross-browser incompatibility?

Two types of articles can be distinguished: those that propose a tool developed by its own authors (17 articles, 54.8%), and those that examine commercial tools (3 articles, 10%). One article (3.3%) proposes a testing technique using an already existing tool (S23). Table 3 lists the tools developed by the studies' authors, these papers propose a testing technique as well.

Table 3 Author developed tools that implement the proposed testing techniques.

Tool	Ref.	Testing technique
Webdiff	(S1, S2, S3)	DOM model analysis, visual analysis
CrossT	(S4)	DOM model analysis, navigation model analysis
CrossCheck	(S6)	DOM model analysis, visual analysis
WebMate	(S7, S11)	DOM model analysis, navigation model analysis
X-PERT	(S10, S13)	DOM model analysis, visual analysis, navigation model analysis
Browserbite	(S12, S22)	Visual analysis
WebSee	(S21)	Visual analysis
Crowdcheck	(S19)	Record/replay, DOM model analysis
Crawljax	(S23)	Navigation model analysis
X-Check	(S26, S29)	Record/replay, DOM model analysis, visual analysis
FieryEye	(S27)	Visual analysis
VISOR	(S31)	Visual analysis

Table 4 Commercial tools evaluated by the articles.

Tool	Observations	Ref.
Adobe Browserlab	Obsolete since March 13, 2013.	(S5)
IE Netrenderer	Only works with Internet Explorer. Free.	
Browsera	Performs full website testing.	
Litmusapp	Obsolete since 2017.	
Browsrcamp	Obsolete.	
IETester	Only works with Internet Explorer.	(S15)
SuperPreview	Part of Expression Web. No longer supported.	
BrowserStack		
BrowserShots	Free.	
CrossBrowserTesting		
Browser Sandbox	Free.	(S25)
IE Tab	Firefox and Chrome extension.	
BrowserCam	Obsolete.	
Browserseal	Obsolete.	

Table 4 lists the examined commercial tools. Some of them are no longer supported, and the others work only on certain configurations.

4.3. RQ3 – How are the proposed tools validated?

The validation methods used by the articles consist on testing the developed tool on websites. In total, 20 articles (66.6%) contain details about the validation process and only 14 of them list the number of test artifacts used. We can classify these articles in those that validate a testing technique (4 papers, 13.3%) as shown in Table 5, and those that validate a tool developed by the authors (10 papers, 33.3%) as shown in Table 6.

Table 5 Number of tested websites on proposed techniques' validation phase.

Technique tested	Selected article	Number of tested sites
Visual analysis	(S17)	4
Visual analysis	(S18)	7
Visual analysis	(S20)	3
Static analysis	(S24)	1

Table 6 Number of tested websites on developed tools' validation phase.

Tool tested	Selected article	Number of tested sites
Webdiff	(S1, S2, S3)	9
Browserbite	(S12, S22)	140
X-PERT	(S13)	14
WebSee	(S21)	8
X-Check	(S26, S29)	8, 11
FieryEye	(S27)	5

In (S13), 10 websites were chosen by the authors. The remaining 4 were obtained by a random URL generator[3]. In (S1, S2, S3) a similar random URL generator was used as well. (S7) has chosen real webpages based on their popularity (Gmail, Craigslist Autos, Virgin America, PayPal). (S20) tested their proposed technique focusing on shopping cart based applications. In (S27), a fault seeding mechanism was used instead of real-world faults. This is because of a lack of access to real refactored web pages (related to visual symptoms implemented in the technique). In (S12, S18, S22, S24), the testing was conducted in the browsers considered the most popular: Google Chrome, Mozilla Firefox, Opera and Internet Explorer.

[3]http://www.uroulette.com/

4.4. RQ 4 - What challenges are presented when implementing the tests approaches?

Fig. 3 shows the most encountered challenges while performing tests. Variable elements detection is one of the most mentioned. It refers to web application regions that are not static. It includes animations, statistics and publicity that change when reloading the website. In (S1, S2, S20), this is handled with a variable region detection strategy. The website is loaded multiple times in the same configuration, and any section of the screen that presents variation is detected. Then, these sections are discarded properly in the tests.

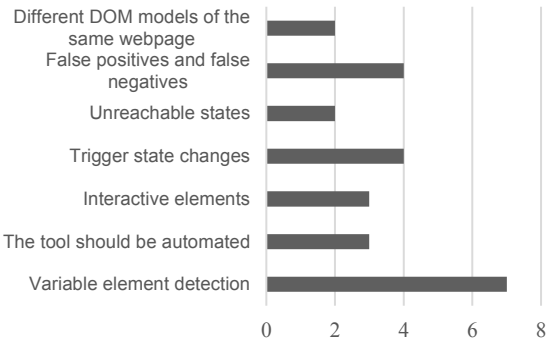

Fig. 3 Challenges found in the articles.

False positive (sometimes called as flaky tests) refers to positive tests that should have resulted negative, because the site is cross-browser compatible. False negative refers to tests that failed despite the presence of incompatibilities. This issue affects all the testing techniques. According to (S6), information gathered from DOM models may produce false positives. Therefore, this technique is used together with visual analysis. In (S12, S22), a classification module for potential incompatibilities in screenshots is detailed. Tests are classified in true positives and false positives. This classification is supported by a machine learning technique using neuronal networks. Machine learning is an artificial intelligence branch that allows computers to learn behavior based on empirical data [16]. Machine learning techniques are used in (S6) as well, to build a more precise visual difference detector. In (S29), it is outlined that navigation model analysis is prone to produce false positives and negatives as well. By crawling the application with web crawlers, non-deterministic actions are ignored.

Triggering state changes is related to navigation model analysis. The articles (S7, S14), mention the difficulty of changing states in the navigation model. When the events are numerous, any click can trigger a new state. In (S16), it is highlighted that performing certain actions in different order may conduct to different states.

Interactive elements are a recurring issue in the selected studies (S7, S16). Web 2.0 technologies promote dynamic behavior with source code that is executed in the client-side, like JavaScript and HTML5 [43].

The need for a tool to be automated is evident in several articles (S6, S9, S29). It is mentioned the high cost associated with the need of a user performing manual actions (S6). However, (S8, S9) propose entirely manual testing techniques.

Unreachable states are related to navigation model analysis. According to (S14, S29), there are states that cannot be reached from the website links. Thus, the produced model will be incomplete, leaving part of the system untested.

Table 7 displays the relation between the most mentioned challenges and the testing techniques discussed.

Table 7 Documented challenges by the selected studies related to the testing techniques.

Challenge	Testing technique
Variable element detection	Visual analysis
Interactive elements	DOM model analysis, Navigation model analysis
Trigger state changes	Navigation model analysis
Unreachable states	Navigation model analysis
Different DOM models of the same webpage	DOM model analysis

Besides the more common challenges mentioned, other issues have been identified. (S2) states that is difficult to take screenshots in visual analysis. This article also exposes the existing issues related to the presence of embedded objects when trying to generate DOM models. For (S3), web browsers that implement security measures constituted a challenge when extracting information to create DOM models. According to (S4), there are cases when an incompatibility is not shown in a DOM model, which leads to false positives. For (S6), sometimes a browser's data provided to construct the DOM model are not accurate, which again, leads to false positives. In (S12), it is described how the variation of parameters needed for visual analysis resulted unproductive, since a reduction in false negatives conducted to an increment of false positives. The authors tried to solve this issue with a neural network model, which achieves high precision at the expense of lower recall. Recall is a term used in machine learning, referring to the proportion of real positive cases that are correctly predicted [44]. Certain CSS properties can cause incompatibilities

(S16), because these may not be fully supported by the browser. A given example is the *expression* property, which only works on Internet Explorer. In (S22), a challenge which may affect any test regardless of the testing technique is discussed: when classifying a potential incompatibility as a real incompatibility (opposite of a false positive), there is a certain degree of subjectivity in which critical and non-critical differences may vary on different applications. They affirm that the proposed tool (Browserbite) is just for static pages, needing modifications to work on adaptive webpages. The static technique proposed in (S24) has a limitation, since the tests are computed on the webpage's source code directly, this should be available to the testers. Finally, in (S27), the use of visual symptoms is proposed when testing. The challenge discussed is the need for a visual symptom to be independent of any webpage. A symptom should cover all issue's source, and to distinguish itself from other related issues' symptoms. Due to the big number of similar properties, the authors argue that this idea is doubtful.

5. Related work

There are few secondary studies related to the SLR developed in this paper. Even though they do not study cross-browser testing specifically, they focus on other types of testing techniques on the UI layer. We mention the studies as follows.

Two studies on web application testing are presented by Garousi et al. [45] and Dogan et al. [9].The first one is a systematic mapping, followed by a deep analysis using a systematic review. These studies were conducted by the same research team.

Paz and Pow-Sang [46] present a systematic review to identify evaluation methods which are employed to assess the usability of applications. The preferred method is usability testing. In the same context, Al-Ismail and Sajeev [47] present a SLR on primary studies related to mobile web usability, user experience and usability challenges for mobile web browsing.

Mascheroni et al. [48] present a study on cross-browser compatibility testing in a continuous software development environment. The same authors also study web incompatibility detection using digital image processing [49].

Saleem et al. [50] propose a systematic review focused on how to maintain quality (based on parameters like reliability, compatibility, etc.), the types of used tests and how to improve quality. The findings have proven that the major contribution for quality assurance of web services, is done by maintaining compatibility issues, the effectiveness of services and traceability of operations.

Garousi et al. [51] conducted a multivocal literature review, which is a type of SLR that includes data from gray literature (such as blog posts, white papers and videos). This study looked for models, challenges and benefits of software test maturity (measured with test maturity assessments (TMA)) and test process improvement (TPI).

Therefore, to our knowledge, this is the first SLR which studies cross-browser compatibility testing.

6. Discussions

6.1. Findings and current research trends

RQ 1- Techniques for cross-browser testing: the visual analysis is the most used technique, as 16 studies selected it. Eight of the authors developed tools that applied this technique, four of those used it in combination with other techniques, and the rest of them used it exclusively. Three studies explored and validated this technique without the development of a proper tool. DOM model analysis is the second most used technique, although there is a decline in its use after 2015. This technique was always implemented in combination with other techniques. The navigation model analysis facilitates the testing of entire websites, and it is also used in combination with other techniques. Other less popular techniques such as record/replay, static analysis, attribute comparison and heuristic evaluation have also been proposed.

RQ2 - Tools: we classified the studied tools in (1) tools developed by the article's authors (12 tools, 46.1%) and (2) commercial tools evaluated by the study (14 tools, 53.8%). Twelve tools were developed and proposed by studies' authors. From these, 7 tools used a combination of testing techniques, while the rest have implemented a single technique. The techniques that have been applied to these tools are: DOM model analysis, visual analysis, navigation model analysis, and record/replay. None of the developed tools used static analysis, attribute comparison or heuristic evaluation. Also, 14 commercial tools were listed, although 6 became obsolete or are no longer supported.

RQ3 - Validation of developed tools and techniques: 20 articles provided validation data, from those, only 14 listed a proper number of test artifacts used on the process. From the 14 articles, 13 use visual analysis as a testing technique (making it the most documented technique on validation phase), 3 use DOM model analysis, 2 use record/replay, static analysis and navigation model analysis are used in 1 article each. Four studies validated a technique and ten validated a tool. The selection of tested sites and web browsers seemed to

be based on popularity. Some authors highlight the difficulty in validation due to the need of unavailable website information.

RQ4 - Challenges presented upon implementation: we listed 7 challenges which were mentioned by several studies. Five of these studies are related to a specific testing technique. The most mentioned challenge was variable element detection, which affects the visual analysis. The other two challenges (false positives and false negatives, and the necessity for the tool to be automated), affect to all the techniques. Other challenges (mentioned only once across the selected article pool) were discussed in the results. These issues are related to how the testing technique was implemented or how the researchers validated the tool.

The research trends found are closely associated to the challenges listed in the RQ4. Several studies (S1, S2, S4, S17, S20, S31) purport the need to improve the tool's precision and decrease the number of false positives by tweaking the configuration variables involved in the testing algorithms. Several studies agree upon the importance in handling variable elements and adaptive websites (S1, S10, S17, S20, S22). Some studies evidence the insufficiency in the validation phase and appraise to increase the number of tested sites (S1, S2, S18). In (S1, S2) it is highlighted the importance to include mobile platforms in the tests. In (S7, S11) it is evidenced the importance of covering server-side parts of the application under test. The studies (S6, S10, S29) suggest investigating techniques to assist in diagnosing and automatic fixing of cross browser incompatibilities through browser-specific automated web page repairs. (S14) delves into benchmarking and security as other possible areas to expand the testing tool's coverage. Some studies (S4, S16, S23), have sufficiently proven the importance of tool automation. In studies that research testing techniques which depend on databases (S4, S24), it is highlighted the importance to increase the available data to improve the results.

6.2. Threats to validity

Here we present potential threats to validity to this study and the actions we took to minimize their impact. A secondary study in threats to validity in SLRs served as guideline for this section [52].

6.2.1. Construct validity

Construct validity is the correct operational measure for the concepts being studied [52]. To avoid not including important resources we used multiple databases to reduce errors during the primary studies searching phase.

To reduce the possibility of incorrect or incomplete search terms, we included key terms synonyms used with logical connectors, along with each search engine's specific configurations to obtain the best results from each search engine.

To avoid using an incorrect search method, we adjusted the search terms according to each repository, performing minor changes to find all relevant sources.

6.2.2. Internal validity

The main threat is an incomplete selection of relevant studies. Our review method is discussed in section 2. To ensure a complete as possible pool of resources, the search was handled in iterations. The search terms were refined to exhaust all possible results that may contain relevant articles. Nevertheless, there is the possibility to have missed pertinent studies to our work. To avoid bias in the study selection, a careful analysis of each article's title, abstract, keywords and conclusions was performed. This method was decided before the selection phase. To prevent article duplication, each article was double checked to discard duplicate results from the repositories search. To minimize subjective quality assessment, we established clear inclusion/exclusion criteria (discussed in section 2).

6.2.3. External validity

The selected articles analyzed for this work were published between 2010 and 2017. This study is within the software engineering scope, the findings and results are only valid in the web application testing field and cross-browser compatibility testing specifically.

6.2.4. Conclusion validity

The data extraction method was guided by the research questions to achieve consistent extraction of relevant information. To ensure the presented results and conclusions are traceable to the data, we presented graphs generated directly from the extracted information. The discussions and conclusions are product of those data. This assures this work can be replicated by others obtaining the same results.

7. Conclusions and future work

Currently, the selected platform for communication and exploitation is the web and effective web testing becomes a challenge for developers. The technology's rapid evolution and its popularity have increased the production of complex and dynamic applications [43, 53]. This makes usability, compatibility and availability, key success attributes

in the web [1]. However, different web browsers produce different content [54, 55, 56]. Thus, the need for cross-browser compatibility testing gains importance.

In this paper, we have studied the developed techniques to test cross-browser compatibility. The visual analysis is the most popular technique. We also have reviewed the proposed tools to implement the tests. The tools were classified depending on their source: developed by the authors or commercial applications.

We also addressed the validation phase of the proposed tools and techniques, listing the articles that provided a concrete number of testing artifacts. The visual analysis is the most validated technique.

Finally, we listed the challenges described in the articles. The presence of dynamic objects was one of the biggest challenges found, especially in modern web applications [53]. The tests automation was an important requirement mentioned as well. Several tools and techniques are developed with different levels of automation. Therefore, it is necessary to automate the testing process.

As future work, we propose the development of a tool to conduct cross-browser compatibility testing, addressing the challenges found in this SLR.

Acknowledgements

This work has been supported by two research projects. One of the projects is "Metodologías y herramientas emergentes para contribuir con la calidad del software" (PI 17F018 SCyT UNNE). The other Project is Análisis e Implementación de tecnologías emergentes en sistemas computacionales de aplicación regional (PI 17F017 SCyT UNNE).

This paper is also part of a project called "Software para la automatización de pruebas de compatibilidad web en un entorno de desarrollo continuo de software" supported by Consejo Interuniversitario Nacional (CIN).

Competing interests

The authors have declared that no competing interests exist.

References

[1] E. Dustin, J. Rashka and D. McDiarmid, *Quality Web Systems: Performance, Security, and Usability*, Boston, Massachusetts, USA: Addison Wesley, 2001.

[2] V. S. Moustakis, C. Litos, A. Dalivigas and L. Tsironis, "Website Quality Assessment Criteria," *IQ,* pp. 59-73, 2004.

[3] J. F. Kurose and K. W. Ross, *Redes de computadoras. Un enfoque descendente*, Pearson, 2010.

[4] "Introduction to cross browser testing – Learn web development | MDN". Available: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/Introduction. Accessed on 2017-04-03.

[5] L.N. Sabaren, M. A. Mascheroni, C. L. Greiner and E. Irrazábal, "Una Revisión Sistemática de la Literatura en Pruebas de Compatibilidad Web," in *XXIII Congreso Argentino de Ciencias de la Computación (CACIC 2017)*, pp. 812-821, La Plata, Argentina, 2017.

[6] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," Keele University and Durham University Joint Report, EBSE 2007-001, 2007.

[7] Z. Zakaria, R. Atan, A. A. A. Ghani and N. F. M. Sani, "Unit Testing Approaches for BPEL: A Systematic Review," in *Software Engineering Conference, APSEC '09. Asia-Pacific*, pp. 316-322, Penang, Malaysia, 2009.

[8] E. Mendes, "A systematic review of Web engineering research," in *International Symposium on Empirical Software Engineering*, pp. 498-507, Queensland, Australia, 2005.

[9] S. Dogan, A. Betin-Can and V. Garousi, "Web application testing: A systematic literature review,"*Journal of Systems and Software,* vol. 91, pp. 174-201, 2014.

[10] E. I. Nabil, "Specifications for Web Services Testing: A Systematic Review," in *2015 IEEE World Congress on Services*, pp. 152-159, New York, USA, 2015.

[11] S. Choudhary, H. Varsee and A. Orso, "A cross-browser web application testing tool," in *2010 IEEE International Conference on Software Maintenance*, pp. 1-6, Romania, 2010.

[12] S. R. Choudhary, H. Versee and A. Orso, "WEBDIFF: Automated Identification of Cross-browser Issues in Web Applications," in *26th IEEE International Conference on Software Maintenance*, pp. 1-10, Romania, 2010.

[13] S. Choudhary, "Detecting Cross-browser Issues in Web Applications," in *2011 33rd International Conference on Software Engineering*, pp. 1146-1148, Hawaii, USA, 2011.

[14] A. Mesbah and M. R. Prasad, "Automated Cross-Browser Compatibility Testing," in *2011 33rd International Conference on Software Engineering*, pp. 561-570, Hawaii, USA, 2011.

[15] J. G. Ochin, "Cross Browser Incompatibility: Reasons and Solutions," *International Journal of Software Engineering & Applications, July 2011,* vol. Vol.2, nº No.3, pp. 66-77, 2011.

[16] S. Choudhary, M. Prasad and A. Orso, "CROSSCHECK: Combining Crawling and Differencing To Better Detect Cross-browser Incompatibilities in Web Applications," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, pp. 171-180, Montreal, Quebec, Canada, 2012.

[17] V. Dallmeier, M. Burger, T. Orth and A. Zeller, "WebMate: A Tool for Testing Web 2.0 Applications," in *Workshop on JavaScript Tools*, pp. 11-15, Beijing, China, 2012.

[18] A. Issa, J. Sillito and V. Garousi, "Visual Testing of Graphical User Interfaces: an Exploratory Study Towards Systematic Definitions and Approaches," in *2012 14th IEEE International Symposium on Web Systems Evolution*, pp. 11-15, Trento, Italy, 2012.

[19] A. Sivaji, N. A. Ramli, Z. M. Nor, N.-K. Chuan, F. Wan, A. Wan and S. Shi-Tzuaan, "Measuring and Improving Website User Experience using UX Methodologies: A Case Study on Cross Browser Compatibility Heuristic," in *Southeast Asian Network of Ergonomics Societies*, pp. 1-6, Langkawi, Kedah, Malasia, 2012.

[20] S. Choudhary, M. Prasad and A. Orso, "X-PERT: Accurate identification of cross-browser issues in web applications," in *2013 35th International Conference on Software Engineering*, pp. 702-711, San Francisco, California, USA, 2013.

[21] V. Dallmeier, M. Burger, T. Orth and A. Zeller, "WebMate: Generating Test Cases for Web 2.0," in *International Conference on Software Quality, Software Quality. Increasing Value in Software and Systems Development*, pp. 55-69, Vienna, Austria, 2013.

[22] N. Semenenko, M. Dumas and T. Saar, "Browserbite: Accurate Cross-Browser Testing via Machine Learning over Image Features," in *2013 29th IEEE International Conference on Software Maintenance*, pp. 528-531, Eindhoven, Netherlands, 2013.

[23] S. Choudhary, M. Prasad and A. Orso, "X-PERT: a web application testing tool for cross-browser inconsistency detection," in *2014 International Symposium on Software Testing and Analysis*, pp. 417-420, San Jose, California, USA, 2014.

[24] A. Deursen, A. Mesbah and A. Nederlof, "Crawl-based analysis of web applications: Prospects and challenges," *Science of Computer Programming,* vol. 87, pp. 173-180, 2014.

[25] B. Kaalra and K. Gowthaman, "Cross Browser Testing Using Automated Test Tools," *International Journal of advanced studies in Computer Science and Engineering,* vol. 3, nº 10, pp. 7-12, 2014.

[26] X. Li and H. Zeng, "Modeling web application for cross-browser compatibility testing," in *2014 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 1-5, Nevada, USA, 2014.

[27] S. Mahajan and W. Halfond, "Finding HTML presentation failures using image comparison techniques," in *29th ACM/IEEE international conference on Automated software engineering*, pp. 91-96, Vasteras, Sweden, 2014.

[28] E. Selay, Z. Q. Zhou and J. Zou, "Adaptive Random Testing for Image Comparison in Regression Web Testing," in *2014 International Conference on Digital lmage Computing: Techniques and Applications*, pp. 1-7, New South Wales, Australia, 2014.

[29] M. He, H. Tang, G. Wu and H. Zhong, "A Crowdsourcing framework for Detecting Cross-Browser Issues in Web Application," in *the 7th Asia-Pacific Symposium on Internetware*, pp. 239-242, Wuhan, China, 2015.

[30] A. Hori, S. Takada, H. Tanno and M. Oinuma, "An oracle based on image comparison for regression testing of web applications," in *27th International Conference on Software Engineering and Knowledge Engineering*, pp. 639-645, Pittsburgh, USA, 2015.

[31] S. Mahajan and W. G. J. Halfond, "Detection and Localization of HTML Presentation Failures Using Computer Vision-Based Techniques," in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation*, pp. 1-10, Graz, Austria, 2015.

[32] T. Saar, M. Dumas, M. Kaljuve and N. Semenenko, "Browserbite: cross-browser testing via image processing," *Software—Practice & Experience,* vol. 46, nº 11, pp. 1459-1477, 2015.

[33] H. Shi and H. Zeng, "Cross-Browser Compatibility Testing Based on Model Comparison," in *2015 International Conference on Computer Application Technologies*, pp. 103-107, Matsue, Japan, 2015.

[34] S. Xu and H. Zeng, "Static Analysis Technique of Cross-Browser Compatibility Detecting," in *2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence*, pp. 103-107, Okayama, Japan, 2015.

[35] N. Barskar and C. Patidar, "A Survey on Cross Browser Inconsistencies in Web Application," *International Journal of Computer Applications,* vol. 137, nº 4, pp. 37-41, 2016.

[36] M. He, G. Wu, H. Tang, W. Chen, J. Wei, H. Zhong and T. Huang, "X-Check: A Novel Cross-browser Testing Service based on Record/Replay," in *2016 IEEE International Conference on Web Services*, pp. 123-130, San Francisco, California, USA, 2016.

[37] S. Mahajan, B. Li, P. Behnamghader and W. G. J. Halfond, "Using Visual Symptoms for Debugging Presentation Failures in Web Applications," in *2016 IEEE International Conference on Software Testing, Verification and Validation*, pp. 191-201, USA, 2016.

[38] M. Sharma and C. P. Patidar, "An Automated Approach for Cross-Browser Inconsistency (XBI) Detection," in *9th Annual ACM India Conference*, pp. 141-145, India, 2016.

[39] G. Wu, M. He, H. Tang and J. Wei, "Detect Cross-Browser Issues for JavaScript-Based Web Applications Based on Record/Replay," in *2016 IEEE International Conference on Software Maintenance and Evolution*, pp. 78-87, Raleigh, North Carolina, USA, 2016.

[40] C. Patidar, M. Sharma and V. Sharda, "Detection of Cross Browser Inconsistency by Comparing Extracted Attributes," *International Journal of Scientific Research in Computer Science and Engineering,* vol. 5, nº 1, pp. 1-6, 2017.

[41] M. Furkan Kıraç, B. Aktemur and H. Sözer, "VISOR: A Fast Image Processing Pipeline with Scaling and Translation Invariance for Test Oracle Automation of Visual Output Systems," in *Journal of Systems and Software*, vol. 136, pp. 266-277, 2018.

[42] "W3C Document Object Model," Available at: https://www.w3.org/DOM/. Accessed on 2017-05-22.

[43] "What Is Web 2.0 - O'Reilly Media," Available at: http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html. Accessed on2017-05-03.

[44] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation,"*Journal of Machine Learning Technologies,* vol. 2, nº 1, pp. 37-63, 2011.

[45] V. Garousi, A. Mesbah, A. Betin-Can and S. Mirshokraie, "A systematic mapping study of web application testing," *Information and Software Technology,* vol. 55, nº 8, pp. 1374-1396, 2013.

[46] F. Paz and J. A. Pow-Sang, "Current Trends in Usability Evaluation Methods: A Systematic Review," in *2014 7th International Conference on Advanced Software Engineering and Its Applications*, pp. 11-15, Haikou, China, 2014.

[47] M. Al-Ismail and A. Sajeev, "Usability challenges in mobile web," in *2014 IEEE International Conference on Communication, Networks and Satellite*, pp. 50-55, Jakarta, Indonesia, 2014.

[48] M. Mascheroni, M. Cogliolo and E. Irrazábal, "Automatización de pruebas de compatibilidad web en un entorno de desarrollo continuo de software," in *Simposio Argentino de Ingeniería de Software - JAIIO 45*, pp. 51-63, 2016.

[49] M. Mascheroni, M. Cogliolo and E. Irrazábal, "Automatic detection of Web Incompatibilities using Digital Image Processing," *Electronic Journal of Informatics and Operations Research,* vol. 16, nº 1, pp. 29-45, 2017.

[50] G. Saleem, F. Azam, M. Younus, N. Ahmed and L. Yong, "Quality assurance of web services: A systematic literature review," in *2016 2nd IEEE International Conference on Computer and Communications*, pp. 1391-1396, Chengdu, China, 2016.

[51] V. Garousi, M. Felderer and T. Hacaloğlu, "What We Know about Software Test Maturity and Test Process Improvement,"*IEEE Software,* vol. 35, nº 1, pp. 84-92, 2017.

[52] X. Zhou, Y. Jin, H. Zhang, S. Li and X. Huang, "A Map of Threats to Validity of Systematic Literature Reviews in Software Engineering," in *2016 23rd Asia-Pacific Software Engineering Conference*, pp. 153-160, Hamilton, New Zealand, 2017.

[53] M. Jazayeri, "Some Trends in Web Application Development," in *Future of Software Engineering 2007, FOSE '07*, pp. 199-213, Minneapolis, Minesota, USA, 2007.

[54] E. Kiciman and B. Livshits, "AjaxScope: a platform for remotely monitoring the client-side behavior of web 2.0 applications," in *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, Stevenson, pp. 17-30, Washington, USA, 2007.

[55] F. Ricca and P. Tonella, "Web testing: a roadmap for the empirical research," in *Seventh IEEE International Symposium on Web Site Evolution*, pp. 63-70, Budapest, Hungary, 2005.

[56] R. Ramler, E. Weippl, M. Winterer, W. Schwinger and J. Altmann, "A Quality-Driven Approach to Web Testing," in *Ibero American Conference on Web Engineering*, pp. 81-95, Santa Fé, Argentina, 2002.