IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 January 2023, accepted 1 February 2023, date of publication 3 February 2023, date of current version 14 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3242549

SURVEY

Methods for Automatic Web Page Layout Testing and Analysis: A Review

IRFAN PRAZINA, (Graduate Student Member, IEEE), ŠEILA BEĆIROVIĆ, (Graduate Student Member, IEEE), EMIR COGO, AND VENSADA OKANOVIĆ

Faculty of Electrical Engineering, University of Sarajevo, 71000 Sarajevo, Bosnia and Herzegovina

Corresponding author: Irfan Prazina (iprazina1@etf.unsa.ba)

ABSTRACT Methods for automatic analysis of user interfaces are essential for a wide range of applications in computer science and software engineering. These methods are used in software security, document archiving, human-computer interaction, software engineering, and data science. Even though these methods are essential, no single research systematically lists most of the methods and their characteristics. This paper aims to give an overview of different solutions and their applications in the separate processes of automatic analysis of user interfaces. The main focus is on the techniques that analyze web page layouts and web page structure. Web pages' style, type of content, and even structure constantly (often drastically) change, as do methods that analyze them. The fact that most methods use very different datasets and web pages of various complexities are some of the reasons that the direct comparison of methods is difficult, if not impossible. Another fact is that the vast applications of methods practically solve similar problems. With these facts in mind, in the paper, we surveyed relevant scientific articles, categorized them, and provided an overview of how these methods have developed over time.

INDEX TERMS GUI similarity detection, GUI testing, HCI, information retrieval, software engineering.

I. INTRODUCTION

The need for automatic analysis of user interfaces originates from different fields of computer science, from computer security to software engineering. In the last 10 to 15 years, many new methods in various areas have been published. However, they have never been systematically analyzed and listed in one place. This fact is the primary motivation for writing this paper. The paper focuses on methods for testing and processing web page layouts. After selecting and finding relevant articles, we noticed many solutions that target similar problems in different fields.

Web-based interfaces are present in a wide variety of applications. Their importance can be seen from the following perspectives:

- **minimal setup** - they are available in web browsers which usually do not require additional setup like in desktop of native mobile applications;
- **portability** - a web-based application that users can access on different kinds of devices. Usually, the only requirement is the internet connection;
- **availability of data** - users can share data in the application with different users, and make queries on a vast amount of data;
- **control of access** - the application owners have a way to provide their application to the users in a way that they can reliably control.

These are some of the factors that make web applications attractive for various fields of application. For example, in the case of software engineering, the exciting aspect of web applications is development and maintenance. On the other hand, in information retrieval, they are essential due to the amount of readily available data, and due to the number of users and various types of users, they are also attractive for research in the field of Human-Computer Interaction.

A web page user interface consists of an HTML code that gives structure and content and a CSS code that defines the

The associate editor coordinating the review of this manuscript and approving it for publication was Mahmoud Elish.

style of each UI element on the page. Many factors must be considered when programming user interfaces for web pages. Some of them are:

- Screen size - users can view a web page on devices with widely different screen sizes. If the web page is designed and implemented for one screen size, it does not mean it will be usable on other screen sizes. Change in screen size can make elements of UI overlap or protrude from the screen;
- Web browser - each web browser can have a different engine for processing and displaying HTML and CSS code. Even though there are standards for CSS and HTML, these differences can cause one web page to be correctly displayed on one web browser and incorrectly on another;
- Type of device or operating systems - Users can use a web page on a wide variety of devices with different configurations and consequently face different problems in the web page layout;
- Localization and internationalization - a web page that is localized in one language can change its appearance. For example, some labels could be longer than on the original page, which can break a web page layout;
- Different configuration of the same web browser - changes in web browser configuration, plugins, and themes can have different effects on the web page appearance;
- Animations and dynamic contents - dynamic changes in content or element animations affect methods that use screenshots comparison. Two screenshots of the same web page can be detected as different web pages when dynamic contents or animations exist on the page;
- Cross effects of CSS rules - one HTML element or component can have CSS code that overrides some common rule on the web page which can change the appearance of the web page when a component or element is added or changed.

These factors are why testing and analysis of web page user interfaces can be a difficult problem, making most methods that work on desktop user interfaces unusable on web pages.

In this paper, an overview of the most influential papers on this topic is given, a chronology of the evolution of the methods, and a list of methods and their approaches to the problem. To the authors' knowledge, this has never been done in one paper. Motivations for methods are different. For example, some methods do phishing detection by analyzing the layout and structure of user interfaces, some do regression testing of web page layouts, some use the layout information as another factor in data analysis, and some test layouts on different devices and web browsers. This paper aims to give a better understanding of which methods exist and highlights the main problems that are being solved. Some methods have been developed with one application goal, possibly without knowing other possible fields where similar problems are considered.

TABLE 1. List of abbreviations.

Abbreviation	Meaning
GUI	Graphical User interface
HTML	Hyper Text Markup Language
XML	Extensible Markup Language
CSS	Cascading Style Sheets
HCI	Human-Computer Interaction
DOM	Document Object Model
SVM	Support Vector Machine
SIFT	Scale-Invariant Feature Transform
ANN	Artificial Neural Networks

During the evaluation, we asked the following questions:

- **RQ1** – What research fields analyze and test web-based user interfaces?
- **RQ2** – What problems are researchers solving by analyzing user interfaces?
- **RQ3** – Is there an overlap of problems solved in separate research areas?
- **RQ4** – What research methodologies are used?
- **RQ5** – Is the field of user interface analysis still active and evolving?

We formulated research questions to cover different areas of research. The first two questions give a comprehensive view to discover many research areas and problems. The significance of **RQ3** and **RQ4** is to provide structure and meaning to results. Finally, the last question highlights the context of the topic in time (past and present).

A. LIST OF ABBREVIATIONS

Multiple abbreviations of names for technologies, terms, and approaches appear in this paper. To avoid the problem of confusion and make it easier to find the meaning of these abbreviations, we created a table of all the abbreviations that appear in this work. Also, explaining abbreviations in the text where they appear can sometimes disrupt the reading flow. In the Table 1 we list all used abbreviations and their meanings.

II. RELATED WORK

This section gives an overview of the papers that survey the problem of automatic web page layout analysis in different fields.

In [1], overview of the papers focused on GUI testing is given. The main conclusion is that most methods are model-based, but in a real-world application, methods such as Selenium, JFCSUnit, and Android Monkey are not based on models. This paper does not give insight into which methods can perform layout analysis in graphical user interfaces.

From the aspect of web security, there is an overview [2] of methods used to detect web-based phishing attacks. Some of the methods mentioned in the paper are based on the visual analysis of the web pages. Therefore, this paper is used as a starting point for finding relevant papers in this field.

Automatic layout analysis is also used in the field of data information retrieval. Overview of those papers can be found

in [3] and [4]. Even though they analyzed many methods, their focus was not only on the layouts of web pages. Many of the methods are focused on different types of documents. Some of the methods overlap with the topic of this paper.

Regarding Human-Computer Interaction, the paper [5] gives an overview of methods for automatic analysis of web interfaces' usability. The authors show that about 33% of methods are automated, and most methods are analytical modeling or simulation-based. This paper gives a detailed and systematic overview which is used as guidance. Since publishing, many new methods have been developed that are not covered by this paper.

To the authors' knowledge, no work has made an extensive analysis and review of methods for automatic analysis of user interfaces or websites in terms of the layout.

III. RESEARCH METHODOLOGY

We used recommendations presented in [6] to write the review. A strategy for selecting and evaluating articles based on the advice is formulated. We divided our research strategy into three steps: searching, filtering, and evaluating papers. The first step's goal is to include as extensive a set of relevant papers as possible. The second step aims to reduce the number of articles so that only those that deal with the topic and are of sufficient quality remain. The last step requires a deeper analysis of the works and a better understanding. We chose this strategy to optimize the time that needs to be spent to consider each paper. After the first two steps, we can study the remaining papers in detail in a reasonable amount of time.

A. SEARCH METHODOLOGY

The research is made using different indexing services for scientific papers. All the papers considered are in the English language and are peer-reviewed. The selection is made in the first stage so that all papers have at least minimal intersection with the topic. Keywords used in the search are ''layout testing'', ''web page similarity'', ''user interface layout testing'', ''user interface similarity'', ''visual GUI testing'', and ''layout faults web page''. Search engines used are Science Direct, Google Scholar, ACM Digital library, and Academic Microsoft (Academic Microsoft was discontinued in the middle of the research). Different search engines are used to obtain a more exhaustive set of papers. After the preliminary search, more than 400 results are found. Papers are then filtered by title and abstract. After that, all papers that do not cover the topic are eliminated from further research.

B. FILTERING METHODOLOGY

All documents found in the first step (search) are filtered by the rules of inclusion and exclusion formulated in Table 2. Some articles do UI testing, which is focused on UI event testing. This type of testing is not part of the central theme of testing and analyzing a layout of UI elements. This exclusion criterion filtered about 80% of papers in this step. Other articles ruled out are not peer-reviewed self-reported articles or short papers. We can make this first filtering step fast by analyzing abstracts and conclusions, and it does not require deep paper analysis.

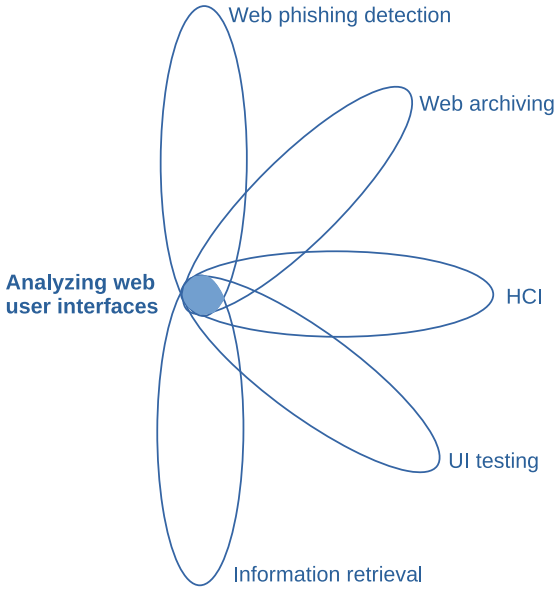


FIGURE 1. Overlap of problems in different fields.

Rules for filtering are made to cover different research areas but in a structured way. All papers need to follow two basic rules:

- The goal of the research in the paper is similar or the same as web page layout testing or analysis
- There are defined methodologies and explanations of the presented method

After filtering 66 papers remained (46 published in conferences, and 20 published in journals). These papers are then analyzed and evaluated in detail following research questions (presented in I).

C. EVALUATION METHODOLOGY

After filtering, there are roughly five areas into which the works can be classified based on their contribution. Different problems and approaches which solve them are recognized in these works, but in all papers, the topic of analysis of (web-based) user interfaces is present. This classification is based on the answer to the **RQ1**.

With this in mind, this paper will give an overview of the main approaches in five domains (Figure 1).

For each domain, we have listed papers that are relevant to it. When examining articles, we paid attention to **RQ2** and **RQ4**. The first part refers to the formulation of the problem and the problem-solving approach, and the second part refers to the experiment's type, description, and results. Some papers do not have quantitative results. But they have valuable observations and recommendations. For those papers, the main conclusions and approaches to the problem are noted and used for overview and comparison.

After reviewing the works of each domain, we present summary statistics on the problems being solved. This

TABLE 2. Inclusion and exclusion criteria for filtering papers.

Inclusion criteria	Exclusion criteria
Peer-reviewed papers	Papers which are not peer-reviewed
Published from 2000 until present	Papers which test only UI events not their layouts
Written in English language	Methods they present are not automatic or at least semi automatic
Have a clear research methodology	Short papers
Are accessible and indexed in search engines stated in search methodology	

information, together with the individual answers to **RQ4**, provides a solution to **RQ3**. By analyzing papers' publication frequency, we can get an answer to **RQ5**.

IV. RESULTS

The automatic analysis of web page layout can be used in many fields of computer science and engineering (Figure 1). In this section, five fields are considered, and their specific problems and solutions are presented. Those fields are:

- Security - Web phishing detection
- Archiving - Web page archiving
- Human Computer Interaction
- Software Engineering - User interface testing
- Information retrieval

The list of the research fields is the answer to the **RQ1**.

We analyzed some statistics for all papers selected after the filtering step in four categories (the ''Web archiving'' category has only one paper, so analysis is trivial). The first statistic is article source type, conference, or journal (see Figure 2). For example, in the categories of ''Web phishing detection'' and ''UI testing'', most papers are published in conferences, and only a few are published in scientific journals. On the other hand, the categories ''Information retrieval'' and ''Human-Computer Interaction'' papers are equally published in conferences and journals.

In the following list, we present the most common journals that include two or more papers (journals are sorted by the number of published papers, all papers that remained after filtering are counted):

- Elsevier - Information and Software Technology
- ACM Computing Surveys
- Elsevier - Data & Knowledge Engineering
- Springer - Empirical Software Engineering
- Wiley - Software Testing, Verification and Reliability

The list of most common conferences is made in the same way:

- IEEE Conference on Software Testing, Validation and Verification
- ACM International Symposium on Software Testing and Analysis
- ACM The Web Conference or formerly known as WWW conference
- IEEE/ACM International Conference on Automated Software Engineering

In another analysis, we examined experiment methodologies in the same categories (see Figure 3). This is an

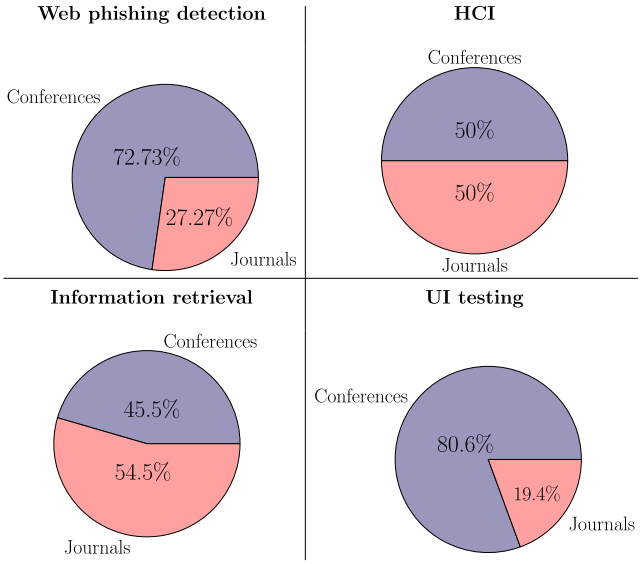


FIGURE 2. Percentage of selected papers by source type (66 papers in total).

answer to the **RQ4**. We recognized four types of experiment methodologies:

- Case-based - experiments are based on a few selected applications or cases. They can produce quantitative or qualitative results, which are usually less statistically significant than experiments that are categorized as qualitative or quantitative
- Qualitative - experiments are qualitative, usually in the form of interviews or questioners and are statistically validated
- Quantitative - experiments are based on larger data sets (more than a few selected cases like in case based approach), and results are numerical
- No experimental data - usually, papers classified in this category are technical papers or reports of the implemented method

This statistic shows that most papers (>57%) have done quantitative experiments. The difference is that the qualitative experiments were present in the ''Human-Computer Interaction'' category in about 30% of all experiments. In other fields, qualitative experiments are less common - the ''UI testing'' category had the second most significant portion of qualitative experiments (14.3%).

While most papers recognize the significance of quantitative experiments, qualitative experiments are not that

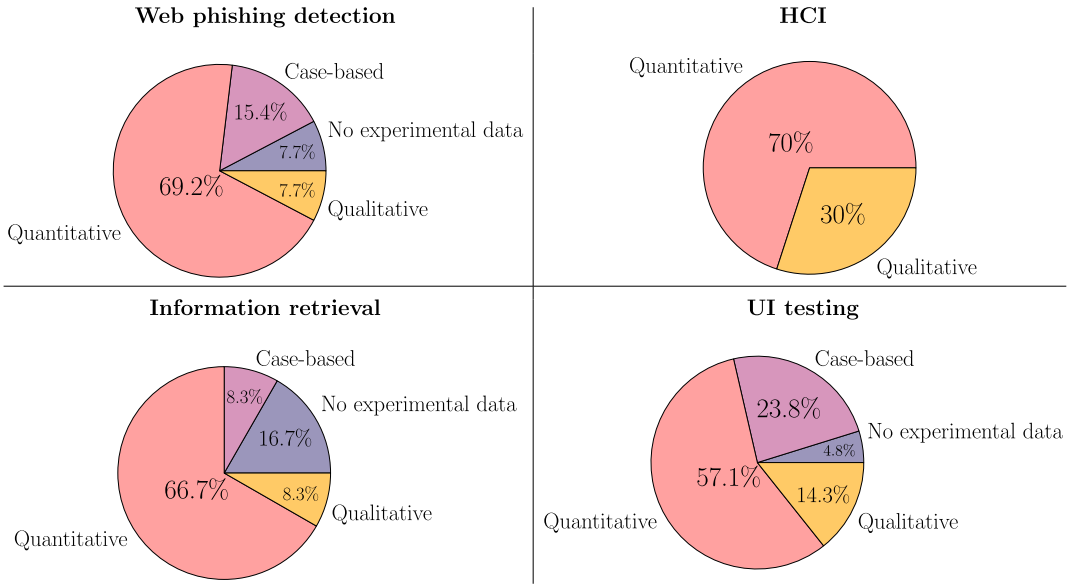


FIGURE 3. Percentage of performed experiments in selected papers (66 papers in total).

popular in all categories. When analyzing experiments not categorized as quantitative or qualitative in the fields of ''UI testing'' and ''Web phishing detection'', we found that many experiments are case-based. These experiments are based on a small set of cases representative of a research question. These experiments can be a good way of showcasing a solution or method presented in a paper but can not replace quantitative experiments. All articles that have just case-based experiments are published at conferences. This observation can be explained by the fact that journals expect more substantial proof of results for the paper to be published.

When we analyzed papers by the year of publishing, we noticed that all categories except UI testing stagnated in the last years - **RQ5** (see Figure 4). Furthermore, in the case of ''UI testing'' we can see the growth of papers published in the previous years. Keeping in mind that these categories solve similar problems, interesting questions arise. For example, can methods or solutions developed in the ''UI testing'' field be applied to other fields? Another question is if the growth we see for UI testing will follow for different areas. The next section will present a more detailed overview and discussion of problems in selected fields.

The paper focuses mainly on Software Engineering because of its growing popularity and the number of new articles in the field. Nevertheless, perspectives of all fields give a complete picture of the topic.

A. WEB PHISHING DETECTION

Phishing is a computer security attack where a malicious user presents a counterfeit web page as legitimate to deceive a user. The user of the fake web page enters confidential data, which will be sent to the malicious user. If this attack is executed right, the user cannot see the difference between a real and fake web page.

In [2], authors say that most attacks of this type target companies that deal with money transactions. The number of these attacks is measured in hundreds of thousands every year.

Papers presented in this section usually use one or combination of the following approaches (Figure 5):

- Analysis of HTML elements and DOM tree - Document Object Model tree is a tree structure used to represent web page content in the web browser memory. The tree depicts HTML elements and their relations as tree nodes and children.
- Visual analysis - This approach uses web page screenshots for image comparison. Screenshots can be influenced by many factors like a web browsing device, a screen, and a web browser
- Analysis of multimedia content - This approach tracks content included on the web page and checks if target content is found on the phishing page.

There are many methods of protection. Some are based on URL analysis, some use search engine data, and some (relevant to this paper) assess the threat by analyzing visual similarity. This method usually looks for the most distinct features found on phishing web pages. However, methods that use web page similarity can be complex, and their results can depend on the type of web browsers and devices a user uses. This is the main problem solved in this section. Answer to the **RQ2** in terms of web phishing detection is solving the problem of assessing web page similarity.

In [7] authors collect the most used CSS selectors and attributes and form a vector that is used to train the SVM classifier. Support Vector Machine (SVM) is a famous machine learning technique often used as a classifier or for data regression. Precision and recall of the method from the paper are above 90%. This paper uses only CSS without

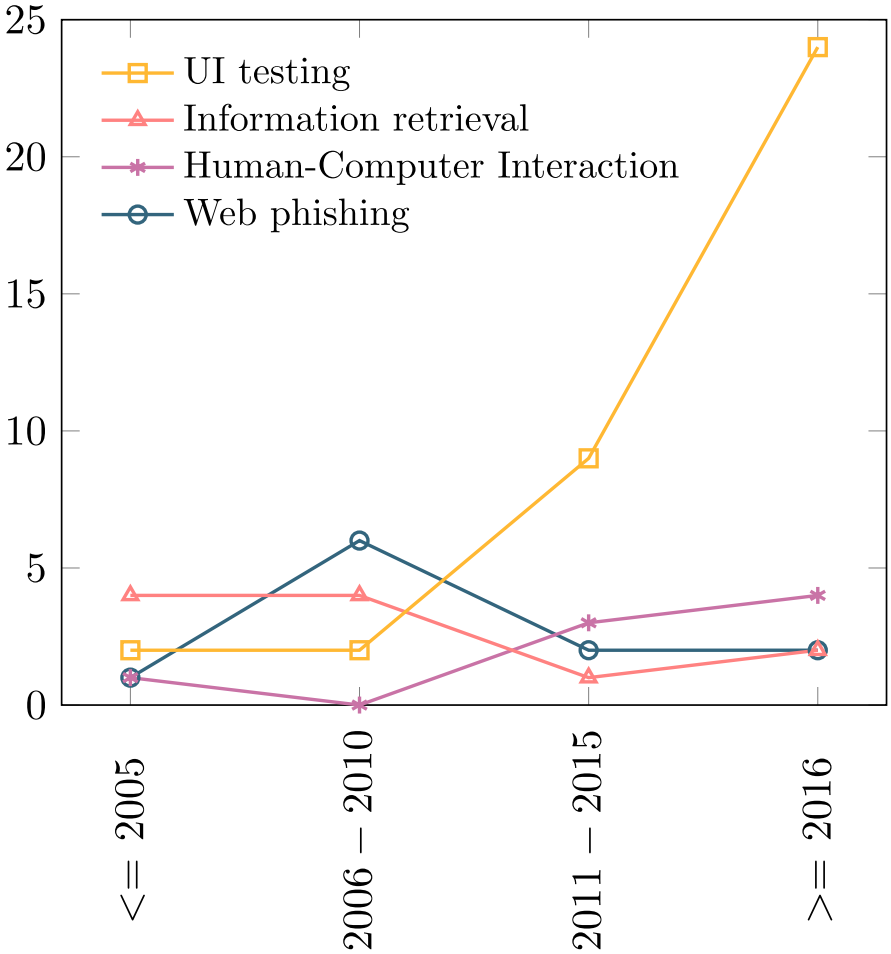


FIGURE 4. Number of published papers in years 2000 – 2022 (66 papers in total).

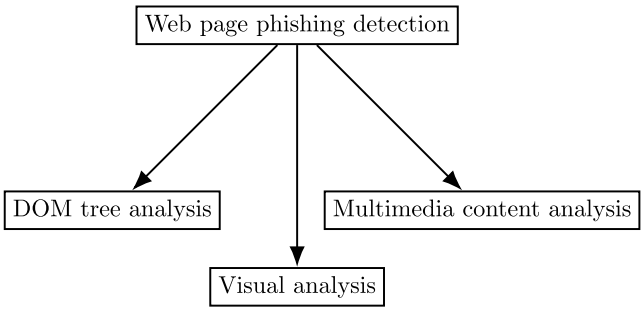


FIGURE 5. Approaches for web phishing detection based on user interface analysis.

visual comparison, which can be problematic if a target web page contains unused CSS code. A more sophisticated approach is described in [8], which is based on a classifier ensemble. The results show that using an ensemble of classifiers boosted detection accuracy up to 15-20% from the basic machine learning classifier.

Many papers use hybrid approaches, a combination of the before mentioned. Papers that combine DOM analysis and visual analysis are [7], [9], [10], [11], and [12]. Mentioned papers do not consider the web page's responsiveness and the different states and configurations of a device and a web browser. These factors can have a significant impact on visual comparison.

Papers [13] and [14] use only information on the frequency of elements in the web page content. They do not analyze elements' relations or properties.

Few papers use HCI principles as similarity metrics for phishing detection. In [15], a metric of perceived similarity for a case study of login screen spoofing is evaluated. The System detects deception with an error rate of 6-13%. A Similar method with more metrics and tests is presented in [16]. The method is based on Gestalt principles. Evaluation of the method is made on four different scenarios. Results show that the approach can consistently discriminate between similar and dissimilar web pages.

A similar problem of phishing is present in mobile applications. The paper [17] presents a plagiarism/phishing detection method based on appearance similarity evaluation. The method shows promising results which are not statistically evaluated (no data on statistical significance and validity). The connection with a web page phishing detection method

TABLE 3. Table of all papers in section - Web phishing detection Type of approach a) DOM tree analysis b) Visual analysis c) Multimedia analysis.

Ref.	First author	Title	Year	Conf.	Journal	Type of approach		
						a)	b)	c)
[7]	J.Mao	Detecting Phishing Websites via Aggregation Analysis of Page Layouts	2018	✓		✓		
[8]	N. Sanglerdsinlapachai	Web phishing detection using classifier ensemble	2010	✓		✓		✓
[9]	L. Wenyin	Detection of phishing webpages based on visual similarity	2005	✓			✓	
[10]	W. Zhang	Web phishing detection based on page spatial layout similarity	2013		✓	✓	✓	
[11]	A. Rosiello	A layout-similarity-based approach for detecting phishing pages	2007	✓		✓		
[12]	W. Liu	An antiphishing strategy based on visual similarity assessment	2006		✓		✓	
[13]	E. Medvet	Visual-similarity-based phishing detection	2008	✓			✓	✓
[14]	M. Hara	Visual Similarity-based Phishing Detection without Victim Site Information	2009	✓				✓
[15]	L. Malisa	Detecting mobile application spoofing attacks by leveraging user visual similarity perception	2017	✓		✓	✓	
[16]	T.C. Chen	Detecting visually similar Web pages: Application to phishing detection	2010		✓		✓	
[17]	J.Zhu	Appearance similarity evaluation for android applications	2015	✓		✓		

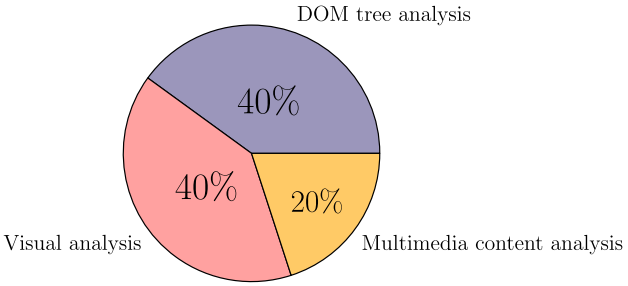


FIGURE 6. Percent of main approaches used in this section (from 11 papers in this category).

is in that this approach is based on the mix of approaches of visual analysis and analysis of UI elements from XML layout files.

The distribution of all approaches in web phishing detection, divided into three categories, can be seen in Figure 6. In web phishing detection, two main approaches are present equally (each 40%). Visual analysis or DOM tree analysis (or elements properties analysis) makes up 80% of all methods. The rest is for multimedia content analysis which does not analyze the whole web page but its multimedia parts.

A summary of all papers discussed in this section and their approaches to solving the problem of web phishing detection is given in Table 3.

B. WEB PAGE ARCHIVING

Website archiving is a way of storing information published on the internet. A lot of this information will disappear with the shutdown of the web page if there are no archives. Initiatives to create such archives have existed since 1996, and 17 petabytes of information are currently archived.

Web archiving is generally done by automated scripts that search the web and store the content of found web pages. It is necessary to check whether the page was archived earlier to avoid archiving redundant data. Difference detection is usually done with a comparison of the visual similarity of the pages. The number of differences is used to decide if we should archive a new page copy. Similar to the previous section, web page archiving solves the problem of assessing web page similarity. This is the answer to the **RQ2** for the category of papers in the web page archiving section.

One paper [18] was found in this category. This paper presents a method for detecting changes on web pages to decide if a new version of the page should be archived. This method uses SVM and is applied over SIFT properties - these properties do not depend on scale. The method is successfully tested on 1000 pages. Although many works from other categories could probably be used in archiving as well, the authors of these papers have not considered this application.

C. HUMAN-COMPUTER INTERACTION

The design of user interfaces should be intuitive and easy to use. Analyzing the layout of elements on the user interface can help designers choose the look of the user interface that is familiar to users so that the position of the elements is in the expected and easily accessible place. Analysis of the layout of the elements on the website can indicate how much visual load the user would experience. The visual load is affected by the number of elements on one page, their mutual relations, and deviations from expected positions. Methods in this category often try to formalize a model of human perception of pages so that they can evaluate the appearance of pages in an objective way.

Another type of problem is web page segmentation. This type of problem is often solved when we need to add some

additional meaning to the elements in the web page (for example, in screen navigation for differently abled persons). The classification of problems can be seen in Figure 7 (this is also the answer to the **RQ2** for papers in the HCI category).

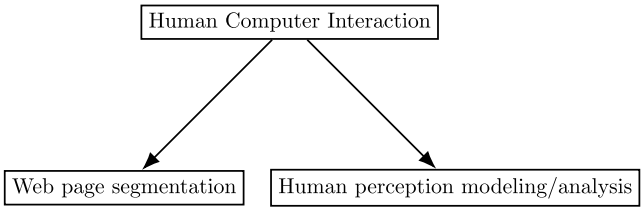


FIGURE 7. Two categories of problems solved in Human-Computer Interaction.

Some methods in this category help people with disabilities use their websites. We have found an example of such work in [19], where the authors propose a method for segmenting web pages based on images. The experiments compare their method with the VIPS image processing tool. The method shows better results than the VIPS tool because this tool is general-purpose, and some modern websites cannot be processed with it. This page segmentation can help navigate between screen reader segments and reduce the visual complexity of pages by eliminating unnecessary things. The authors state that this method could also be used for generating website sketches.

The following paper that performs website segmentation is [20]. In this paper, page segmentation is done to determine the position on the page that will be suitable for inserting an advertisement. The difference between this and previous work is that segmentation, in this case, is done only with text as an input parameter, while previous work considers the page's visual representation. The proposed method shows good results, but it would be good to see if combining visual information and text would contribute to better results.

Papers [21] and [22] deal with how people perceive web pages. While paper [22] provides recommendations for conducting manual comparisons of website similarities, [21] provides a formalized method in the form of an algorithm. Unlike the vast majority of papers, [21] has a detailed description and availability of a data set and easily reproducible experiments. We can see the use of Artificial Neural Networks in papers [23] and [24]. The paper [23] uses readily available metrics in ANN to assess web page similarity without employing actual users. More detailed experiments and approaches that combine visual similarity and neural networks can be seen in [24]. Even though [24] seems comprehensible, it does not use structural information from web pages.

Analyzing user interfaces is done similarly for mobile applications [25]. The method detects common user interface design patterns in Android applications by analyzing static XML files. Mobile applications need to be disassembled to obtain XML files, unlike web pages where HTML and CSS are available from the start.

In [26], the authors present an experiment in which they offer users different variants of spatial transformations of the user interface and measure how quickly they can get used to such changes. They do this to discover which principles of the spatial arrangement of elements should be used in the design of user interfaces. This paper concludes that scaling has the most negligible impact, while the process of getting used to the changed interface is slower for moving elements into a new row and making the page longer. The limitation of this paper is the relatively small number of participants in the experiment.

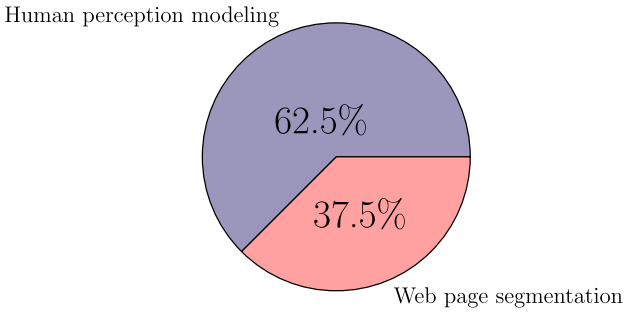


FIGURE 8. Percent of papers in each problem's category solved in HCI (from 8 papers in this category).

The main problem solved in HCI which uses user interface data from web pages, is human perception modeling/analysis. We found this problem in 62.5% of papers in this section. The rest of the papers solve the problem of web page segmentation. This is illustrated in Figure 8.

Some papers in this category use methods used in other areas, primarily web page segmentation. The drawback is that most of them do not compare their results with these methods.

A summary of all papers discussed in this section and their approaches to solving the problem of Human-Computer Interaction is given in Table 4.

D. INFORMATION RETRIEVAL

The amount of information on the Internet is practically limitless. For them to be helpful, we need methods to find relevant information from this large set. Some of this information is in the form of websites. Methods that find relevant information are focused on several factors, including analysis of textual content and analysis of the structural appearance of the document. This section presents methods that analyze the arrangement of elements on web pages when calculating similarities between web pages.

In Figure 9, we can see the main problems in the field of information retrieval where web page user interface analysis is performed. These problems are answers to the **RQ2** from the point of view of information retrieval. More about problems and their solutions can be found in this section.

Some of the methods in this category, like some of the previously mentioned, aim to detect similar pages. In [27], a method based on bipartite graph pairing is presented, where changes on web pages are detected by analyzing the tag

TABLE 4. Table of all papers in section - Human-Computer Interaction Type of approach a) Web page segmentation b) Human perception modeling.

Ref.	First author	Title	Year	Conf.	Journal	Type of approach	
						a)	b)
[19]	M. Cormier	Purely vision-based segmentation of web pages for assistive technology	2016		✓	✓	
[20]	Z. Wu	Position-wise contextual advertising: Placing relevant ads at appropriate positions of a web page	2013		✓	✓	
[21]	A. Bozkir	Layout-based computation of web page similarity ranks	2018		✓		✓
[22]	G. Martine	That site looks 88.46% familiar: quantifying similarity of Web page design	2005		✓	✓	
[23]	M. Bakaev	Evaluation of user-subjective Web interface similarity with Kansei engineering-based ANN	2017	✓			✓
[24]	M. Bakaev	Assessing Similarity for Case-Based Web User Interface Design	2018	✓			✓
[25]	A. S. Shirazi	Insights into layout patterns of mobile user interfaces by an automatic analysis of android apps	2013	✓			✓
[26]	J. Scarr	Testing the robustness and performance of spatially consistent interfaces	2013	✓			✓

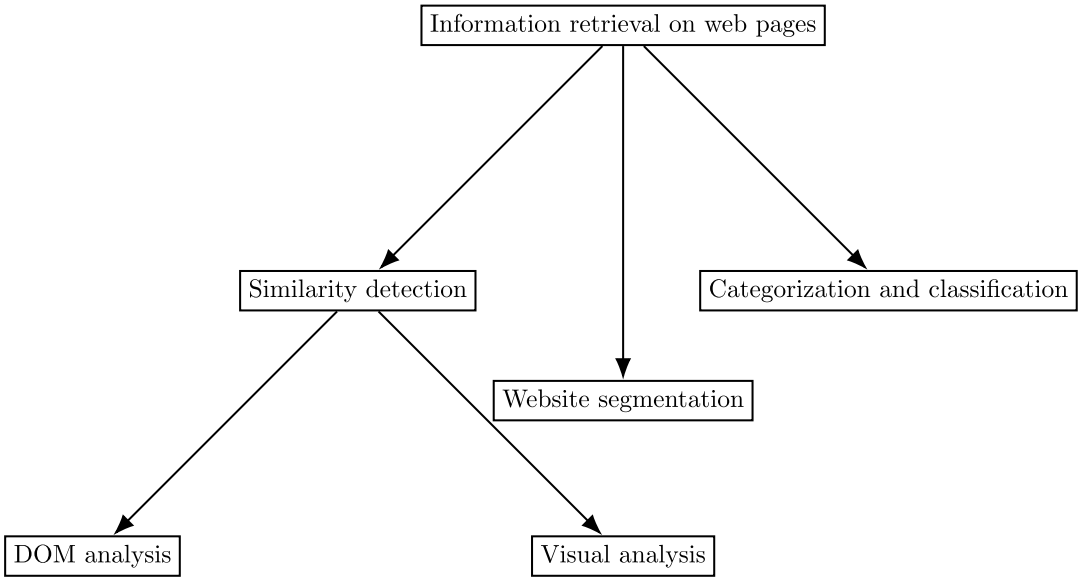


FIGURE 9. Main categories of problems solved in information retrieval on web pages.

structure of HTML documents. In [28], the authors present a similar method, but unlike the previous paper, there is no detailed description of the experiment or test data. Another similarity metric is the frequency of tags appearing in an HTML document. This metric is used in paper [29]. The disadvantage of the experiments in this paper is that the test data set is relatively old. In addition to analyzing HTML tags, metrics for similarity can be visual information such as element edges. One method that uses such information is presented in [30]. In their experiments on their dataset, the authors demonstrated that their method delivers better results than methods that only use color information to recognize similarities between websites.

Website segmentation is an essential step in data processing for information retrieval. In [31], segmentation is performed based on visual content. The results of the experiments confirm that the method is fast and accurate, but the disadvantage of this experiment is that it did not use the actual web browser environment but a library that represents an abstraction and simplification of how a web browser behaves. This can affect the results because the implementation of the library used deviates from the environment of web browsers used by users. The importance of layout information of websites in information retrieval can be seen in papers [32], [33], and [34]. The paper [32] uses the properties of div and table DOM elements. The paper concludes that hybrid methods give better results than segmentation using only structural information. Models for learning the importance of blocks within web pages based on neural networks and SVM are presented in [33]. The main conclusions in the paper are that people have consistent decisions about which blocks are essential and that, in addition to spatial features, implies that better results can be obtained by integrating the content properties of the pages.

Finally, the paper [34] presents a prototype tool that uses information about the layout of elements on web pages in the form of images. The disadvantage of the last paper is that the tool has not been adequately tested on actual data.

In addition to segmentation for information retrieval, categorization or classification of web pages is also important. In [35], the authors present a method based on neural networks that analyzes the images included in the page and determines which category of pages belongs. Another paper [36], which is relatively older, classifies web pages using information from the DOM tree. Due to different test data and the way the experiments were conducted, the results of these works are impossible to compare. Moreover, both papers do not use available layout information in the website classification process.

Information on websites can be found using query-based language. One such language is presented in [37]. However, the experiment performed in the paper is limited to only one page and does not give quantitative results.

In this paper's information retrieval category, we recognized two main problems: website segmentation and similarity detection (Figure 10). These two problems are almost equally present in selected papers, making up above 80% of all problems solved in these papers. The rest of the methods solve issues of categorization and classification.
A summary of all papers discussed in this section and their approaches to solving the problem of information retrieval in web pages is given in Table 5.

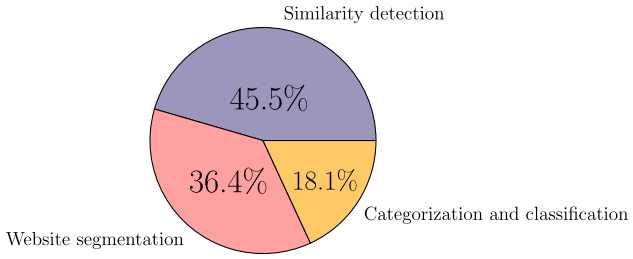


FIGURE 10. Percent of main problems in information retrieval (from 11 papers in this category).

E. SOFTWARE ENGINEERING - TESTING

One of the central stages in software development is testing. Graphical user interface testing is often based on user-triggered events and checks on the consequent application state changes. An aspect that can be tested is the state of the arrangement of the elements. User interfaces of web pages usually consist of several components. Therefore, regression testing is essential for component-based development and agile software development. This type of testing checks the status of the application after a change is made to ensure that it does not generate new errors in the system.

Many methods, which are reviewed below, test the state of the user interface of web applications in various conditions. Part of them deals with testing after changes to individual components. All these tests are done because the style specification using CSS is not interpreted the same on different web browsers, in different environments, and the mutual influence of the style of individual components can affect the final appearance of the website. Pages with an inconsistent display of elements can make it challenging to use and sometimes lead the user to make mistakes. Taking everything into account, below we list the methods that test the user interface of websites by various application categories.

In [38], the authors make recommendations based on the empirical observations that GUI tests should not change unit tests. The speed of execution of GUI tests is comparable to the speed of user actions. In contrast, unit tests are performed at speeds that are close to the processing time that a computer can do. The disadvantage of this research, although with valuable recommendations, is that they did not conduct an experiment to prove some of their claims. On the other hand, in [39], authors notice in their empirical study on visual GUI testing that developing new tests is costlier than maintenance, but also that frequent maintenance is less costly than infrequent. This fact means that these tests need to be resilient to changes. The paper [40] covers the problem of the fragility of visual GUI tests. They stated that around 20%-30% of the test methods had to be modified at least once during the evolution of the app. To solve this problem, the authors in [41] made a proof of concept library that makes test cases independent of the internal structure of UIs. The drawback of this paper is a small dataset. In a paper [42], a method for generating new test scenarios for widgets and test specifications is presented. One of the drawbacks is that the method is semiautomatic - testers need to specify user interactions. Another aspect of testing is developers' perception of the tools they use to make GUI tests. In [43] detailed study on one tool and framework for user evaluation is presented. Conclusions point to the fact that manuals and documentation need to be rich and descriptive, and after the learning curve is passed, users have confidence in writing tests using the described tool. The most significant challenge of maintaining an end-to-end UI test for testing web applications is the fragility of web element locators [44]. This observation is made using mutation analysis and is also per [41] recommendations.

An essential aspect of testing user interfaces are methods that are based on image comparison. In [45], the authors use computer vision to automate testing user interfaces that change as software evolves. The method can detect previously described visual changes, but the disadvantage is that it cannot detect unexpected changes. The other paper [46] uses convolutional neural networks to detect user interface widgets. Using this approach in random testing significantly improved the test code coverage of 18/20 applications on which the method was evaluated. Performance, when compared to solutions that have perfect widget information, indicates that there is potential for improvement. Random testing is also researched in [47], where an approach for adaptive random testing of web pages based on image

TABLE 5. Table of all papers in section - Information retrieval Type of approach a) Similarity detection b) Website segmentation c) Categorization and Classification.

Ref.	First author	Title	Year	Conf	Journal	Type of approach		
						a)	b)	c)
[27]	H. Artail	A fast HTML web page change detection approach based on hashing and reducing the number of similarity computations	2008		✓	✓		
[28]	M. Alpuente	A Visual Technique for Web Pages Comparison	2009		✓	✓		
[29]	A. Tombros	Factors affecting web page similarity	2005	✓		✓		
[30]	Y. Takama	Visual similarity comparison for Web page retrieval	2005	✓		✓		
[31]	J. Zeleny	Box clustering segmentation: A new method for vision-based web page preprocessing	2017		✓		✓	
[32]	G. Hattori	Robust web page segmentation for mobile terminal using content-distances and page layout information	2007	✓			✓	
[33]	R. Song	Learning important models for web page blocks based on layout and content analysis	2004	✓			✓	
[34]	H. Yan	Document page retrieval based on geometric layout features	2013	✓		✓		
[35]	D. Lopez-Sanchez	"Visual content-based web page categorization with deep transfer learning and metric learning"	2019		✓			✓
[36]	V. Crescenzi	Clustering Web pages based on their structure	2005		✓			✓
[37]	G. Della Penna	Visual extraction of information from web pages	2010		✓		✓	

sequence comparison is presented. Experiments are limited to one resolution and show that automating the verification of test output can be made with some help of domain knowledge. Finally, we have found another paper [48] that describes a method for automated detection and localization of presentation failures in web pages. The method was evaluated on four real-world applications and could detect 100% of presentation failures. One drawback that many image processing methods have is a pixel-perfect match which is not always desirable, especially in responsive web pages. In responsive web pages, elements' dimensions and positions depend on the window width. This fact implies that the same web page can result in falsely detected errors if the comparison is done to the presentation on a browser with a slightly different window width or even on the same widow width but with added default margins that can differ from browser to browser.

The program testability needs to be assessed to keep a program code more maintainable. Different test metrics are used for this purpose. In [49], the authors list different testing metrics for user interfaces. The proposed metrics in the paper are consistent and can help assess the complexity of user interfaces. However, the paper would be complete if the metrics were evaluated on a more significant number of projects. In [50], they examine the influence of distance metrics on the perceived similarity of web user interfaces and conclude that the distance of elements on the interface does not affect perceived similarity. The disadvantage of this paper is the relatively small set of test data and participants in the experiment.

Another vital aspect of great software is reusable code. Developing reusable code involves regression testing. This testing verifies whether modifying or creating new functionality introduces errors into the system. There are papers dealing with the regression testing of user interfaces [51], [52]. In [51], the authors offer a method for automatically determining reusability and repairing user interface tests. The paper is based on a control-flow graph, and the implemented tool is evaluated on genuine software. The evaluation shows that the tool can efficiently and effectively fix the tests to be reusable in case of regression. The paper [52] is more concerned with regression testing of web pages. The method analyzes the layout of web pages using layout graphs on two versions of the web page (before and after the change of functionality). The tool developed in this paper was evaluated on 15 actual pages and shows, with only one false positive result, an excellent ability to detect changes in the layout of web pages. The layout graph is a data representation of a web page where elements are represented as nodes and their layout relations as edges. Some graph implementations also capture elements' behavior concerning screen size (responsiveness).

In [53] and [54], the authors present declarative language and a tool for testing the layout of web pages. The justification for introducing a new declarative language is that CSS is a descriptive language that is interpreted in various environments. Differences in interpretations can be due to various factors such as type and version of web browser, type of client device, screen size and resolution, and others. As an outcome of these two papers, we have an empirical analysis and classification of various layout errors and a tool for testing web pages. Papers would be better if they included comparisons of results with other methods. The authors of [55] also present their specification language, which models a set of conditions that user interfaces need to meet. In the experiments, we can see that the paper can successfully detect errors. The paper's disadvantage is that it does not include a comparison with other similar methods.

Paper [56] presents a tool for detecting inconsistencies between page views on different web browsers. The tool is based on web crawling and methods for creating element alignment graphs. It shows promising results in detecting inconsistencies. In addition to detection, [57], [58] also solves the problem of fixing errors due to different page presentations on different web browsers. Both papers in the presented experiments show the success of error correction in 86% of cases.

Another type of error that can occur is related to the localization of websites. The paper [59] describes an automatic technique for detecting errors in web pages' appearance that are caused by internationalization. The method is based on a layout graph and has been tested on 54 different pages. It shows promising results in the detection of this type of error. The average detection time on the test system was 9.75 seconds. A method has been developed to correct this type of error [60]. The solution is made using linear programming. It has been evaluated on 23 actual pages and shows that it can effectively correct the mentioned errors.

As well as errors caused by different web browsers and the process of internationalization and localization, the type of errors that occur due to flaws in the implementation of website responsiveness is also essential. Numerous methods for detecting errors in responsiveness have been developed, and below, we list the most significant and their results. The paper [61] describes a method for automatically checking the layout of web pages for responsiveness errors. An empirical study on about 100 websites, which are obtained using modification operators, shows results for a recall of 91% with 15 false-negative samples. The method is based on layout graphs and the Selenium environment. Selenium environment is open source software used for test automation for web pages.

A portion of the errors in responsive web pages can occur due to how the user interface is implemented so that parts and components are dynamically generated using JavaScript programming language. Automatic detection of visual errors on pages using the user interfaces state graph is described in [62]. This approach solves the problem of detecting errors due to inconsistencies between web browsers and errors caused by dynamic changes in user interfaces using JavaScript. The paper was evaluated with previous work and showed the ability to detect different types of errors. Previous work cannot detect errors due to dynamic changes.

Another method has been developed for detecting errors in dynamic web pages [63]. And this method is also based on layout graphs. The disadvantage of this work is that the results are compared only with the method based on the image comparison. The paper [64] tries to solve the problem of detecting false-positive results in the errors of responsive websites. The method was evaluated on web pages generated with modification operators. The results show that there are no false-positive results. A similar method was developed in [65] but was evaluated on a set of 26 actual pages. The experiment showed that the method on the pages could detect a total of 33 different errors. In addition to detection, it is sometimes necessary to show a visual verification of the errors found. This problem is solved in [66]. The experiments compare manual verification with automatic verification. In conclusion, automatic verification is superior because it can verify the error in less than one second. On the other hand, assessing the quality of the comparison is problematic. Some of the errors humans reject because they are not significant enough. The authors illustrate this with an example: ''For instance, consider an element A that is overlapping the coordinates of an element B, with n pixels of element A overlapping n pixels of B. In this case, a human would decide whether the n pixels of overlap are negligible and if the overall aesthetics remain satisfactory. Both of these criteria are not easily defined and remain, to a great extent, subjective'' [66]

In addition to visual verification, an automatic classification of errors in responsive web pages was done [67]. Based on the performed experiments, the authors claim that the tool developed in the paper can detect different classes of errors with an accuracy of 78.6 % when compared with manual classification.

All previous methods that detect errors in responsive web pages are based on the layout graph. In addition to this approach, the approach described in [68] is based on mathematical logic. The results of the experiments state that the tool can detect errors in less than 5 seconds, but the disadvantage is that the method has not been tested on different web browsers.

Papers [69] and [70] describe different methods that give automatic arrangements of user interface elements. There are a few differences between these methods. Unlike [70], the authors of [69] offer a method that generates multiple arrangements for the same user interface. The paper [70] method uses a mathematical basis for calculating the conditions that should be valid between the elements.

Formal logic is also used to verify and model web page layouts. In paper [71], the formalization of a substantial fragment of the CSS semantics is presented. The model shows promising results when compared with real-life web browsers. Results show few disagreements which originate from rounding errors in Firefox. Verification of web page layouts is done in papers [72] and [73]. The paper [72] presents a method for layout proofs based on formal logic. The paper has eight proofs, and experiments are based on qualitative experience. Formal logic is used in [73] to assess the accessibility of web page layouts. Accessibility is assessed on 62 web pages using assertions based on usability guidelines and standards. In the results, 64 errors are found and only 13 false-positive results.

A summary of all papers discussed in this section and their approaches to solving the problem of UI testing is given in Table 6. The majority of papers in this section solve the problem of testing, with a few of them trying to incorporate human perception modeling. These problems are answers to

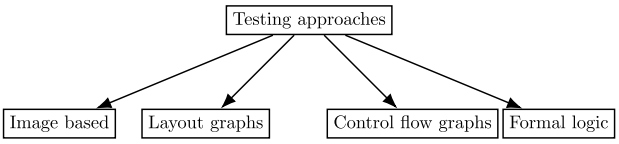


FIGURE 11. Approaches in testing web pages.

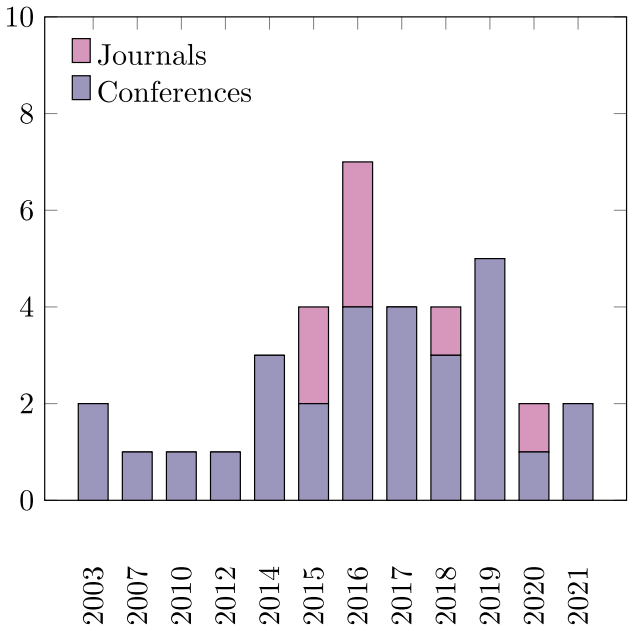


FIGURE 12. Type of publication (36 papers).

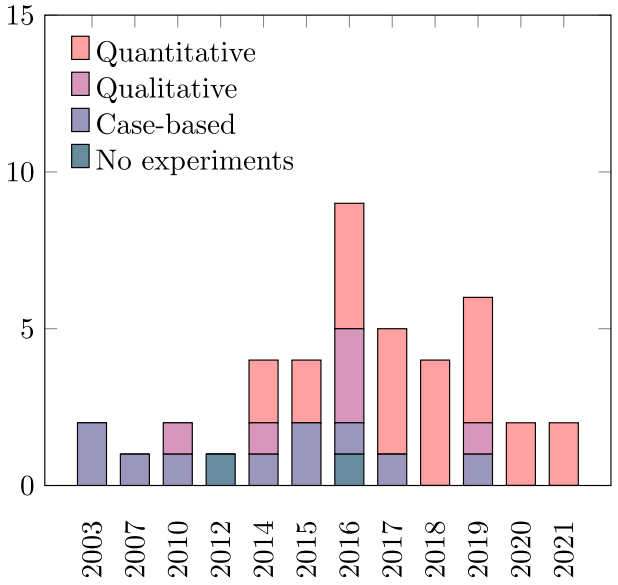


FIGURE 13. Type of conducted experiments in 36 papers from Software Engineering - Testing category.

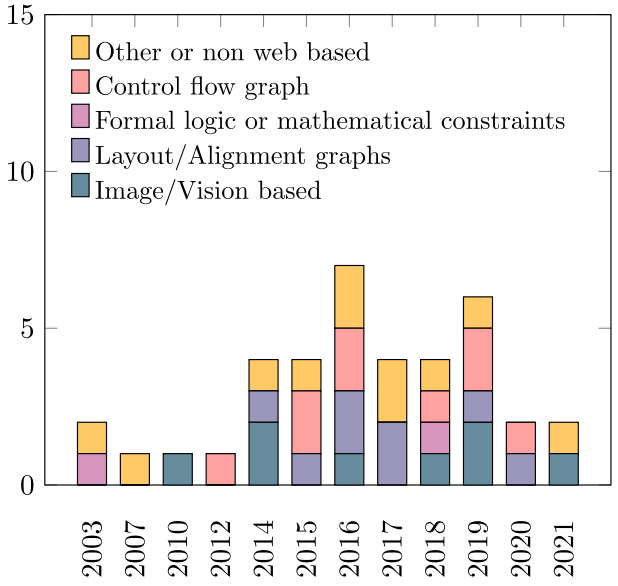


FIGURE 14. Type of method used for testing (36 papers).

the **RQ2**. Different approaches to testing user interfaces in web pages, which are found in papers in this section, can be seen in Figure 11.

1) STATISTICS OF SELECTED PAPERS

All articles (36 in total) classified in this section are analyzed based on three approaches:

- Type of publication - conference or journal
- Type of conducted experiments
- Type of method used to solve the problem

This information is essential for answering **RQ5**, which focuses on the evolution and importance of the topic in the context of time. Papers are classified by year based on each classification approach to present the development of methods in the field of research.

The number of selected articles in software engineering justifies the detailed analysis. This field of study has the most significant number of documents that analyze web page user interfaces. Because of this fact, we can make observations year by year. We could not do this type of analysis in previous sections because papers are spread evenly across years, and the number of documents each year is not larger than two or three.

Most of the works collected in this section have been published at conferences. However, in the last few years, some papers have been published in journals (Figure 12). The number of articles published in a year grew over time; in 2016, that number peaked. After that, the number stagnates. These statistics indicate that the topic is relevant, which is the answer to the **RQ5**.

We presented experimental methods for each paper in Figure 13. Some of the papers have multiple types of experiments. That is why the number of experiments in the chart does not equal the number of papers. Most early papers (up to 2016) have made case-based or qualitative experiments. After the year 2016, most of the papers adopted a quantitative approach. In most cases, the quantitative approach included data sets of artificially modified web pages

TABLE 6. Table of all papers in section - Software engineering Type of approach a) Image based b) Layout/Alignment graphs c) Control flow graphs d) Formal logic e) Other.

Ref.	First author	Title	Year	Conf.	Journ.	Type of approach				
						a)	b)	c)	d)	e)
[38]	C. Lowell	Successful automation of GUI driven acceptance testing	2003	✓						✓
[39]	E. Alegroth	Maintenance of automated test suites in industry: An empirical study on Visual GUI Testing	2016		✓	✓				
[40]	R. Coppola	Fragility of layout-based and visual GUI test scripts: an assessment study on a hybrid mobile application	2019	✓						✓
[41]	H. Pirzadeh	Resilient user interface level tests	2014	✓						✓
[42]	D. T. Dinh	A method for Automated User Interface Testing of Windows-based Applications	2018	✓						✓
[43]	T. Vos	Testar: Tool support for test automation at the user interface level	2015		✓					✓
[44]	R. Yandrapally	Mutation Analysis for Assessing End-to-End Web Tests	2021	✓						✓
[46]	T. White	Improving Random GUI Testing with Image-Based Widget Detection	2019	✓		✓				
[47]	E. Selay	Adaptive random testing in detecting layout faults of web applications	2018		✓	✓				
[48]	S. Mahajan	Finding HTML presentation failures using image comparison techniques	2014	✓		✓				
[49]	K. Magel	GUI structural metrics and testability testing	2007	✓						✓
[50]	S. Heil	Measuring and ensuring similarity of user interfaces: the impact of web layou	2016	✓						✓
[51]	M. Atif	Regression Testing of GUIs	2003	✓				✓		
[52]	T. Walsh	Automatically identifying potential regressions in the layout of responsive web pages	2020		✓		✓			
[53]	S. Halle	Declarative layout constraints for testing web applications	2016		✓				✓	
[54]	S. Halle	Testing web applications through layout constraints	2015	✓					✓	
[55]	F. Zaraket	Guicop: Specification-based gui testing	2012	✓					✓	
[56]	S.R. Choudhary	X-PERT: a web application testing tool for cross-browser inconsistency detection.	2014	✓		✓	✓			
[57]	S. Mahajan	Xfix: an automated tool for the repair of layout cross browser issues	2017	✓						✓
[58]	S. Mahajan	Automated repair of layout cross browser issues using search-based techniques	2017	✓						✓
[59]	A. Alameer	Detecting and localizing internationalization presentation failures in web applications	2016	✓			✓			
[60]	A. Alameer	Efficiently repairing internationalization presentation failures by solving layout constraints	2019	✓					✓	
[61]	T. Walsh	ReDeCheck: an automatic layout failure checking tool for responsively designed web pages	2017	✓			✓			
[62]	Y. Ryou	Automatic detection of visibility faults by layout changes in HTML5 web pages	2018	✓				✓		
[63]	M. Moyeen	An Automatic Layout Faults Detection Technique in Responsive Web Pages Considering JavaScript Defined Dynamic Layouts	2016	✓			✓			
[64]	T. Walsh	Automatic Detection of Potential Layout Faults Following Changes to Responsive Web Pages	2015	✓			✓			
[65]	T. Walsh	Automated layout failure detection for responsive web pages without an explicit oracle	2017	✓			✓			
[66]	I. Althomali	Automatic visual verification of layout failures in responsively designed web pages	2019	✓		✓	✓			
[67]	I. Althomali	Automated visual classification of DOM-based presentation failure reports for responsive web pages	2021	✓		✓				
[68]	O. Beroual	Detecting responsive web design bugs with declarative specifications	2020	✓					✓	
[69]	O.S. Ramon	A layout inference algorithm for Graphical User Interfaces	2016		✓					✓
[70]	N. Jamil	Hildreth's algorithm with applications to soft constraints for user interface layout	2015		✓				✓	
[71]	P. Panchekha	Automated reasoning for web page layout	2016	✓					✓	
[72]	P. Panchekha	Modular Verification of Web Page Layout	2019	✓					✓	
[73]	P. Panchekha	Verifying that web pages have accessible layout	2018	✓					✓	

or some arbitrary list of real-life web pages. Unfortunately, most web pages in use do not have enough testable errors. Therefore, approaches that create multiple versions of a web page with induced errors are used.

In Figure 14, we classified papers by type of method used for testing web page user interfaces. As shown in figure 14, the type of approaches fluctuates over time. In the beginning, most of the methods were image-based, control flow graph-based, or based on formal logic. However, after 2014, layout graph-based techniques were introduced and became dominant or equal to image and control flow graph-based techniques. One of the explanations for this phenomenon is the advent of responsive web pages and mobile-first web pages whose layout is not constant. These observations answer the part of the question **RQ5**, which covers the evolution of the topic in time.

V. DISCUSSION AND CONCLUSION

Many methods for analyzing and testing the user interface of websites have been created recently. This paper provides an overview of these methods' five most important fields. This classification of approaches is the answer to the **RQ1**. However, until this paper, to the authors' knowledge, not a single research considered the automatic analysis of web-based user interfaces from the different aspects of diverse research fields.

A lack of comparison of methods' performances could be a study limitation. However, this aspect of research could not have been done because of the different data sets used in papers, their availability, different methodologies, and the lack of publicly available implementations of published approaches. Therefore, even without performance comparison, we give observations that could be used in the future to mitigate these shortcomings.

This discussion aims to overview findings for each field of study, and outline implied connections between them. In summary, most of the methods solve some of the following problems:

- Web page similarity detection - methods for detecting web phishing attacks notice the similarity between pages and the database of malicious pages. In software engineering, similarity detection is used in regression testing, where it is checked whether a modification in a web page causes an unwanted change in its appearance. There is a similar problem with archiving a website. If a new page appears, is it the same or similar to a page from the archive? This problem is also crucial if you are searching for information from similar pages.
- Segmentation of web pages - by segmenting the website, we recognize individual components and can assign them specific roles or recognize their function within the page. Methods for segmentation are also used in information retrieval and the HCI field.
- Classification and categorization of web pages - classification and categorization based on the appearance of a web page, groups pages into classes that we can use in the fields: information retrieval - to speed up or focus searches, testing - to identify similar errors, HCI - to identify patterns of similar designs.
- Modeling human perception - this problem is dominantly present in the HCI field, but it can also be helpful in testing to assess whether a change on a web page is visible to humans.

We can see that multiple research fields have solutions for each problem. This finding positively answers the **RQ3**, which implies that similar problems are solved in numerous research fields. Unfortunately, in the case of articles reviewed in this paper, there is very little cooperation between different research areas.

We believe developing new methods for analyzing user interfaces in one area can also affect other fields. The lack of mutual referencing and use of techniques from other domains tells us that most scientists are unaware of methods developed in different areas of research interest. The cooperation of various fields could produce better results. One of the examples is the visual verification of errors. Having a real user verify each test result can sometimes make automatic testing pointless. In this process, human perception is essential, which can be modeled using techniques from the HCI research field.

This paper presents the most significant works from five fields and compares their problems, techniques, and solutions. In the future, we would like to see methods from one area be used in other areas. Collecting the answers for **RQ2** and **RQ4**, we saw a lot of similarities between the fields, which gives us additional hope that some discoveries made in one area will, in the future, find their place in other fields as well.

Additionally, the eventual creation of uniform datasets we could use in future works would significantly improve cross-field methods, leading to the fact that we can easily compare the developed solutions. We believe that the presented problem has a vast development space and that most of the potential has yet to be fully reached. When we got statistics related to **RQ5** in Software Engineering - Testing, we noticed that the research area was evolving and developing. We hope that this trend will follow in other areas as well.

REFERENCES

[1] I. Banerjee, B. Nguyen, V. Garousi, and A. Memon, "Graphical user interface (GUI) testing: Systematic mapping and repository," *Inf. Softw. Technol.*, vol. 55, no. 10, pp. 1679–1694, Oct. 2013.
[2] G. Varshney, M. Misra, and P. K. Atrey, "A survey and classification of web phishing detection schemes," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 6266–6284, Dec. 2016.
[3] S. Marinai, B. Miotti, and G. Soda, *Digital Libraries and Document Image Retrieval Techniques: A Survey*. Berlin, Germany: Springer, 2011, pp. 181–204.
[4] F. Alaei, A. Alaei, U. Pal, and M. Blumenstein, "A comparative study of different texture features for document image retrieval," *Exp. Syst. Appl.*, vol. 121, pp. 97–114, May 2019.
[5] M. Y. Ivory and M. A. Hearst, "The state of the art in automating usability evaluation of user interfaces," *ACM Comput. Surveys*, vol. 33, no. 4, pp. 470–516, Dec. 2001.

[6] B. A. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele Univ., Durham Univ., Durham, U.K., Tech. Rep. EBSE 2007-001, Jul. 2007. [Online]. Available: https://www.elsevier.com/__data/promis_misc/525444systematicreviews guide.pdf

[7] J. Mao, J. Bian, W. Tian, S. Zhu, T. Wei, A. Li, and Z. Liang, "Detecting phishing websites via aggregation analysis of page layouts," *Proc. Comput. Sci.*, vol. 129, pp. 224–230, Jan. 2018.

[8] N. Sanglerdsinlapachai and A. Rungsawang, "Web phishing detection using classifier ensemble," in *Proc. 12th Int. Conf. Inf. Integr. Web-Based Appl. Services*. New York, NY, USA: ACM Press, Nov. 2010, pp. 210–215.

[9] L. Wenyin, G. Huang, L. Xiaoyue, Z. Min, and X. Deng, "Detection of phishing webpages based on visual similarity," in *Proc. Special Interest Tracks Posters 14th Int. Conf. World Wide Web (WWW)*. New York, NY, USA: ACM Press, 2005, pp. 1060–1061.

[10] W. Zhang, "Web phishing detection based on page spatial layout similarity," *Informatica*, vol. 37, no. 3, pp. 1–14, 2013.

[11] A. P. E. Rosiello, E. Kirda, C. Kruegel, and F. Ferrandi, "A layout-similarity-based approach for detecting phishing pages," in *Proc. 3rd Int. Conf. Secur. Privacy Commun. Netw. Workshops SecureComm*, 2007, pp. 454–463.

[12] W. Liu, X. Deng, G. Huang, and A. Y. Fu, "An antiphishing strategy based on visual similarity assessment," *IEEE Internet Comput.*, vol. 10, no. 2, pp. 58–65, Mar. 2006.

[13] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netowrks*. New York, NY, USA: ACM Press, Sep. 2008, pp. 1–6.

[14] M. Hara, A. Yamada, and Y. Miyake, "Visual similarity-based phishing detection without victim site information," in *Proc. IEEE Symp. Comput. Intell. Cyber Secur.*, Mar. 2009, pp. 30–36.

[15] L. Malisa, K. Kostiainen, and S. Capkun, "Detecting mobile application spoofing attacks by leveraging user visual similarity perception," in *Proc. 7th ACM Conf. Data Appl. Secur. Privacy*, Mar. 2017, pp. 289–300.

[16] T.-C. Chen, S. Dick, and J. Miller, "Detecting visually similar web pages: Application to phishing detection," *ACM Trans. Internet Technol.*, vol. 10, no. 2, pp. 1–38, May 2010.

[17] J. Zhu, Z. Wu, Z. Guan, and Z. Chen, "Appearance similarity evaluation for Android applications," in *Proc. 7th Int. Conf. Adv. Comput. Intell. (ICACI)*, Mar. 2015, pp. 323–328.

[18] M. T. Law, C. S. Gutierrez, N. Thome, and S. Gancarski, "Structural and visual similarity learning for web page archiving," in *Proc. 10th Int. Workshop Content-Based Multimedia Indexing (CBMI)*, Jun. 2012, pp. 1–6.

[19] M. Cormier, K. Moffatt, R. Cohen, and R. Mann, "Purely vision-based segmentation of web pages for assistive technology," *Comput. Vis. Image Understand.*, vol. 148, pp. 46–66, Jul. 2016.

[20] Z. Wu, G. Xu, C. Lu, E. Chen, Y. Zhang, and H. Zhang, "Position-wise contextual advertising: Placing relevant ads at appropriate positions of a web page," *Neurocomputing*, vol. 120, pp. 524–535, Nov. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231213005110

[21] A. S. Bozkir and E. Akcapinar Sezer, "Layout-based computation of web page similarity ranks," *Int. J. Hum.-Comput. Stud.*, vol. 110, pp. 95–114, Feb. 2018.

[22] G. Martine and G. Rugg, "That site looks 88.46% familiar: Quantifying similarity of web page design," *Expert Syst.*, vol. 22, no. 3, pp. 115–120, Jul. 2005.

[23] M. Bakaev, V. Khvorostov, S. Heil, and M. Gaedke, "Evaluation of user-subjective web interface similarity with kansei engineering-based ANN," in *Proc. IEEE 25th Int. Requirements Eng. Conf. Workshops (REW)*, Sep. 2017, pp. 125–131.

[24] M. Bakaev, *Assessing Similarity for Case-Based Web User Interface Design*. Cham, Switzerland: Springer, 2018, pp. 353–365.

[25] A. Sahami Shirazi, N. Henze, A. Schmidt, R. Goldberg, B. Schmidt, and H. Schmauder, "Insights into layout patterns of mobile user interfaces by an automatic analysis of Android apps," in *Proc. 5th ACM SIGCHI Symp. Eng. Interact. Comput. Syst. (EICS)*. New York, NY, USA: ACM Press, 2013, pp. 275–284.

[26] J. Scarr, A. Cockburn, C. Gutwin, and S. Malacria, "Testing the robustness and performance of spatially consistent interfaces," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, Apr. 2013, pp. 3139–3148.

[27] H. Artail and K. Fawaz, "A fast HTML web page change detection approach based on hashing and reducing the number of similarity computations," *Data Knowl. Eng.*, vol. 66, no. 2, pp. 326–337, Aug. 2008.

[28] M. Alpuente and D. Romero, "A visual technique for web pages comparison," *Electron. Notes Theor. Comput. Sci.*, vol. 235, pp. 3–18, Apr. 2009.

[29] A. Tombros and Z. Ali, *Factors Affecting Web Page Similarity*. Berlin, Germany: Springer, 2005, pp. 487–501.

[30] Y. Takama and N. Mitsuhashi, "Visual similarity comparison for web page retrieval," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, 2005, pp. 301–304.

[31] J. Zeleny, R. Burget, and J. Zendulka, "Box clustering segmentation: A new method for vision-based web page preprocessing," *Inf. Process. Manage.*, vol. 53, no. 3, pp. 735–750, May 2017.

[32] G. Hattori, K. Hoashi, K. Matsumoto, and F. Sugaya, "Robust web page segmentation for mobile terminal using content-distances and page layout information," in *Proc. 16th Int. Conf. World Wide Web (WWW)*. New York, NY, USA: ACM Press, 2007, pp. 361–370.

[33] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma, "Learning important models for web page blocks based on layout and content analysis," *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 2, pp. 14–23, Dec. 2004.

[34] H. Yan and T. Watanabe, "Document page retrieval based on geometric layout features," in *Proc. 7th Int. Conf. Ubiquitous Inf. Manage. Commun. (ICUIMC)*, vol. 60. New York, NY, USA: ACM Press, 2013, pp. 1–8.

[35] D. López-Sánchez, A. G. Arrieta, and J. M. Corchado, "Visual content-based web page categorization with deep transfer learning and metric learning," *Neurocomputing*, vol. 338, pp. 418–431, Apr. 2019.

[36] V. Crescenzi, P. Merialdo, and P. Missier, "Clustering web pages based on their structure," *Data Knowl. Eng.*, vol. 54, no. 3, pp. 279–299, Sep. 2005.

[37] G. Della Penna, D. Magazzeni, and S. Orefice, "Visual extraction of information from web pages," *J. Vis. Lang. Comput.*, vol. 21, no. 1, pp. 23–32, Feb. 2010.

[38] C. Lowell and J. Stell-Smith, *Successful Automation of GUI Driven Acceptance Testing*. Berlin, Germany: Springer, 2003, pp. 331–333.

[39] E. Alégroth, R. Feldt, and P. Kolström, "Maintenance of automated test suites in industry: An empirical study on visual GUI testing," *Inf. Softw. Technol.*, vol. 73, pp. 66–80, May 2016.

[40] R. Coppola, L. Ardito, and M. Torchiano, "Fragility of layout-based and visual GUI test scripts: An assessment study on a hybrid mobile application," in *Proc. 10th ACM SIGSOFT Int. Workshop Automating Test Case Design, Selection, Eval.* New York, NY, USA: ACM, Aug. 2019, pp. 28–34.

[41] H. Pirzadeh and S. Shanian, "Resilient user interface level tests," in *Proc. 29th ACM/IEEE Int. Conf. Automated Softw. Eng.* New York, NY, USA: ACM, Sep. 2014, pp. 683–688.

[42] D. T. Dinh, N. P. Hung, and T. N. Duy, "A method for automated user interface testing of windows-based applications," in *Proc. 9th Int. Symp. Inf. Commun. Technol.* New York, NY, USA: Association for Computing Machinery, 2018, pp. 337–343.

[43] T. Vos, P. Kruse, N. Condori-Fernańdez, S. Bauersfeld, and J. Wegener, "Testar: Tool support for test automation at the user interface level," *Int. J. Inf. Syst. Model. Design*, vol. 6, no. 3, pp. 46–83, Jul. 2015.

[44] R. Yandrapally and A. Mesbah, "Mutation analysis for assessing end-to-end web tests," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2021, pp. 183–194.

[45] T.-H. Chang, T. Yeh, and R. Miller, "GUI testing using computer vision," in *Proc. 28th Int. Conf. Human Factors Comput. Syst. (CHI)*. New York, NY, USA: ACM Press, 2010, pp. 1535–1544.

[46] T. White, G. Fraser, and G. Brown, "Improving random GUI testing with image-based widget detection," in *Proc. 28th ACM SIGSOFT Int. Symp. Softw. Test. Anal.* New York, NY, USA: ACM, Jul. 2019, pp. 307–317.

[47] E. Selay, Z. Q. Zhou, T. Y. Chen, and F.-C. Kuo, "Adaptive random testing in detecting layout faults of web applications," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 28, no. 10, pp. 1399–1428, Oct. 2018.

[48] S. Mahajan and W. G. J. Halfond, "Finding HTML presentation failures using image comparison techniques," in *Proc. 29th ACM/IEEE Int. Conf. Automated Softw. Eng.*, Sep. 2014, pp. 91–96.

[49] K. Magel and I. Alsmadi, "Gui structural metrics and testability testing," in *Proc. Conf. Softw. Eng. Appl.*, 2007, pp. 91–95.

[50] S. Heil, M. Bakaev, and M. Gaedke, "Measuring and ensuring similarity of user interfaces: The impact of web layout," in *Proc. Web Inf. Syst. Eng. (WISE)*. Cham, Switzerland: Springer, 2016, pp. 252–260.

[51] M. Atif and M. L. Soffa, "Regression testing of GUIs," in *Proc. 9th Eur. Softw. Eng. Conf. Held Jointly 11th ACM SIGSOFT Int. Symp. Found. Softw. Eng. (ESEC/FSE)*. New York, NY, USA: Association for Computing Machinery, 2003, pp. 118–127.

[52] T. A. Walsh, G. M. Kapfhammer, and P. Mcminn, "Automatically identifying potential regressions in the layout of responsive web pages," *Softw. Test., Verification Rel.*, vol. 30, no. 6, pp. 183–194, Aug. 2020.

[53] S. Hallé, N. Bergeron, F. Guérin, G. L. Breton, and O. Beroual, "Declarative layout constraints for testing web applications," *J. Log. Algebr. Methods Program.*, vol. 85, no. 5, pp. 737–758, Aug. 2016.

[54] S. Halle, N. Bergeron, F. Guerin, and G. L. Breton, *Testing Web Applications Through Layout Constraints*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Apr. 2015.

[55] F. Zaraket, W. Masri, M. Adam, D. Hammoud, R. Hamzeh, R. Farhat, E. Khamissi, and J. Noujaim, "GUICOP: Specification-based GUI testing," in *Proc. IEEE 5th Int. Conf. Softw. Test., Verification Validation*, Apr. 2012, pp. 747–751.

[56] S. Roy Choudhary, M. Prasad, and A. Orso, "X-PERT: A web application testing tool for cross-browser inconsistency detection," in *Proc. Int. Symp. Softw. Test. Anal. (ISSTA)*. New York, NY, USA: ACM Press, 2014, pp. 417–420.

[57] S. Mahajan, A. Alameer, P. Mcminn, and W. Halfond, "XFix: An automated tool for the repair of layout cross browser issues," in *Proc. 26th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2017, pp. 368–371.

[58] S. Mahajan, A. Alameer, P. Mcminn, and W. G. J. Halfond, "Automated repair of layout cross browser issues using search-based techniques," in *Proc. 26th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2017, pp. 249–260.

[59] A. Alameer, S. Mahajan, and W. G. J. Halfond, "Detecting and localizing internationalization presentation failures in web applications," in *Proc. IEEE Int. Conf. Softw. Test., Verification Validation (ICST)*, Apr. 2016, pp. 202–212.

[60] A. Alameer, P. T. Chiou, and W. G. J. Halfond, "Efficiently repairing internationalization presentation failures by solving layout constraints," in *Proc. 12th IEEE Conf. Softw. Test., Validation Verification (ICST)*, Apr. 2019, pp. 172–182.

[61] T. A. Walsh, G. M. Kapfhammer, and P. Mcminn, "ReDeCheck: An automatic layout failure checking tool for responsively designed web pages," in *Proc. 26th ACM SIGSOFT Int. Symp. Softw. Test. Anal.* New York, NY, USA: ACM, Jul. 2017, pp. 360–363.

[62] Y. Ryou and S. Ryu, "Automatic detection of visibility faults by layout changes in HTML5 web pages," in *Proc. IEEE 11th Int. Conf. Softw. Test., Verification Validation (ICST)*, Apr. 2018, pp. 182–192.

[63] M. A. Moyeen, G. G. M. N. Ali, P. H. J. Chong, and N. Islam, "An automatic layout faults detection technique in responsive web pages considering Javascript defined dynamic layouts," in *Proc. 3rd Int. Conf. Electr. Eng. Inf. Commun. Technol. (ICEEICT)*, Sep. 2016, pp. 1–5.

[64] T. A. Walsh, P. Mcminn, and G. M. Kapfhammer, "Automatic detection of potential layout faults following changes to responsive web pages (N)," in *Proc. 30th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2015, pp. 709–714.

[65] T. A. Walsh, G. M. Kapfhammer, and P. Mcminn, "Automated layout failure detection for responsive web pages without an explicit Oracle," in *Proc. 26th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2017, pp. 192–202.

[66] I. Althomali, G. M. Kapfhammer, and P. Mcminn, "Automatic visual verification of layout failures in responsively designed web pages," in *Proc. 12th IEEE Conf. Softw. Test., Validation Verification (ICST)*, Apr. 2019, pp. 183–193.

[67] I. Althomali, G. M. Kapfhammer, and P. Mcminn, "Automated visual classification of DOM-based presentation failure reports for responsive web pages," *Softw. Test., Verification Rel.*, vol. 31, no. 4, p. e1756, Jun. 2021.

[68] O. Beroual, F. Guérin, and S. Hallé, *Detecting Responsive Web Design Bugs With Declarative Specifications*. Cham, Switzerland: Springer, Jun. 2020, pp. 3–18.

[69] Ó. Sánchez Ramón, J. Sánchez Cuadrado, J. García Molina, and J. Vanderdonckt, "A layout inference algorithm for graphical user interfaces," *Inf. Softw. Technol.*, vol. 70, pp. 155–175, Feb. 2016.

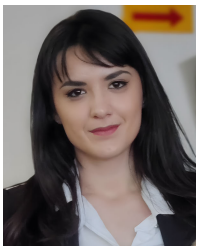
[70] N. Jamil, X. Chen, and A. Cloninger, "Hildreth's algorithm with applications to soft constraints for user interface layout," *J. Comput. Appl. Math.*, vol. 288, pp. 193–202, Nov. 2015.

[71] P. Panchekha and E. Torlak, "Automated reasoning for web page layout," in *Proc. ACM SIGPLAN Int. Conf. Object-Oriented Program., Syst., Lang., Appl.*, Oct. 2016, pp. 181–194.

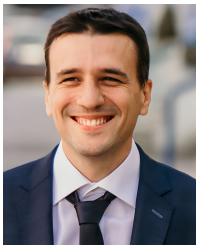
[72] P. Panchekha, M. Ernst, Z. Tatlock, and S. Kamil, "Modular verification of web page layout," in *Proc. ACM Program. Lang.*, vol. 3, Oct. 2019, pp. 1–26.

[73] P. Panchekha, A. Geller, M. Ernst, Z. Tatlock, and S. Kamil, "Verifying that web pages have accessible layout," in *Proc. 39th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, Jun. 2018, vol. 53, no. 4, pp. 1–14.

IRFAN PRAZINA (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees from the Department of Computer Science and Informatics, Faculty of Electrical Engineering, University of Sarajevo, Bosnia and Herzegovina, in 2013 and 2015, respectively, where he is currently pursuing the Ph.D. degree. He is currently a Senior Teaching Assistant with the Department of Computer Science and Informatics, Faculty of Electrical Engineering, University of Sarajevo. His research interests include web technologies, software testing, and mobile application development.



ŠEILA BEĆIROVIĆ (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees from the Department of Computer Science and Informatics, Faculty of Electrical Engineering, University of Sarajevo, Bosnia and Herzegovina, in 2017 and 2019, respectively, where she is currently pursuing the Ph.D. degree. She is currently a Teaching Assistant with the Department of Computer Science and Informatics, Faculty of Electrical Engineering, University of Sarajevo. Her research interests include computer networks and security, mobile application development, and operational research.



EMIR COGO received the B.Sc. and M.Sc. degrees from the Department of Computer Science and Informatics, Faculty of Electrical Engineering, University of Sarajevo, Bosnia and Herzegovina, in 2011 and 2013, respectively, where he is currently pursuing the Ph.D. degree. He is currently a Senior Teaching Assistant with the Department of Computer Science and Informatics, Faculty of Electrical Engineering, University of Sarajevo. His research interests include game development, computer graphics, and procedural modeling.

VENSADA OKANOVIĆ received the B.Sc., M.Sc., and Ph.D. degrees from the Faculty of Electrical Engineering, Sarajevo. She is currently an Associate Professor with the Faculty of Electrical Engineering, Sarajevo. She is also an Active Collaborator with the Sarajevo Graphics Group. She is the coauthor of their projects and publications. Her expertise and research interests include web and mobile applications development, software engineering, web technologies, and graphics programming.