1 Introduction

The appearance of a web application’s UI plays an important part in its success. Studies have shown that users form judgments about the trustworthiness and reliability of a company based on the visual appearance of its web pages, and that issues degrading the visual consistency and aesthetics of a web page have a negative impact on an end user’s perception of the website and the quality of the services that it delivers. Presentation failures can also cause serious usability problems leading to a frustrating user experience. Presentation failures are found to occur frequently in modern web applications — for example, in a recent study over 75% of users reported problems in accessing websites from their mobile devices.

Despite the importance of presentation failures, their detection and repair poses numerous challenges for developers. First, their detection is an expensive task, for example, in the case of XBIs a web page needs to be checked on a large number of browsers to ensure appearance and behavior consistency. Second, localization of presentation failures is difficult given the complex layouts and styles of modern web pages that make it difficult to identify the UI elements responsible for the observed presentation failure. Third, there exist no standardized ways to repair presentation failures, requiring the developers to resolve them on a case by case basis. Fourth, for a repair, developers must modify the problematic UI elements precisely without introducing new presentation failures. Predictably, these challenges have made presentation failures an ongoing topic of concern for developers. For example, a simple search on StackOverflow — a popular technical forum — with the search term “cross browser” results in over 23,000 posts discussing ways to resolve XBIs, of which approximately 7,000 are currently active questions. Existing UI testing techniques are only able to detect and localize presentation failures (i.e., they address the first two of the four previously listed challenges). Repairing them is, hitherto, a manual task that is labor intensive and requires significant expertise.

To address these limitations, my Ph.D. thesis develops a novel general-purpose framework that uses search-based techniques for the automated repair of presentation failures in web applications. The framework abstracts the commonalities and provides points of specialization catering to the different types of presentation failures. The framework introduces a new paradigm for repairing UI related problems in web applications. To evaluate the effectiveness of the framework, I designed and developed its instantiations for repairing different types of presentation failures in web applications, namely, Cross-browser Issues (XBIs), Mobile Friendly Problems (MFPs), Internationalization Presentation Failures (IPFs), Mockup Driven Development Problems, and Regression Debugging Problems. The instantiations have demonstrated high effectiveness in repairing different types of presentation failures in web pages while maintaining their aesthetic quality.
2 Current Work

In this section I discuss the different instantiations I designed for repairing the different types of presentation failures mentioned above.

2.1 \textit{X}Fix: Repair of Cross Browser Issues (XBIs)

A consistent cross-browser user experience is crucial for the success of a website. Differences in how various browsers interpret HTML and CSS standards can result in \textit{Cross Browser Issues (XBIs)} — inconsistencies in the appearance or behavior of a website across different browsers. Tool support to help developers debug XBIs is limited in terms of capabilities. To address these limitations, I have designed a novel search-based approach, \textit{X}Fix, that enables the automated repair of XBIs in web pages [1, 2]. My key insight is that the impact of XBIs can be quantified by a fitness function capable of guiding a search to a repair that minimizes the number of XBIs present in a page. The intuition is that a good fitness function can be built considering two objectives. The first objective leverages a measurement of the number of XBIs detected in a page, by using well-known XBI detection techniques. The second objective measures the similarity of the layout of the page when rendered in different browsers, by comparing the size and positions of the bounding boxes of the HTML elements involved in each XBI identified. The empirical evaluation of \textit{X}Fix on 15 real world web pages showed that it was able to resolve 86% of the XBIs reported by X-PERT, a well-known XBI detection tool, and 99% of the XBIs observed by humans. The results therefore demonstrate that my approach is potentially of high use to developers by providing automated fixes for problematic web pages involving XBIs. This work received the ACM SIGSOFT Distinguished Paper Award at ISSTA 2017, one of the top two international testing conferences.

2.2 \textit{M}Fix: Repair of Mobile Friendly Problems

A large number of websites are not designed to gracefully handle users who are accessing their pages through a mobile device. These sites may exhibit a range of \textit{mobile friendly problems}, such as unreadable text, cluttered navigation, or content that overflows the devices viewport and forces the user to pan and zoom the page in order to access content. Moreover, as of April 2015, Google has incorporated mobile-friendliness as part of its ranking criteria when returning search results to mobile devices. Fixing mobile friendly problems is challenging as it typically requires adjusting dozens of HTML elements and CSS properties in synchronization with each other while at the same time ensuring that they do not impact other parts of the page. To address this problem I designed an approach, \textit{M}Fix, to automatically generate CSS patches that can improve the mobile friendliness of a web page [3]. To do this my approach builds graph-based models of the layout of a web page and uses constraints encoded by these graphs to compute repair patches that can improve mobile friendliness while minimizing layout disruption. The empirical evaluation of \textit{M}Fix on 38 popular websites listed in the Alexa Top 50 most visited websites showed that it could effectively resolve mobile friendly problems for 95% of the subjects. I also evaluated our results with a user study, in which participants overwhelmingly preferred the repaired version of the website for use on mobile devices, and also considered the repaired page to be more readable than the original.

2.3 \textit{I}Fix: Repair of Internationalization Presentation Failures (IPFs)

Internationalization (i18n) is a mechanism that enables companies to effectively communicate with a global audience by providing translated text or localized media content on their websites. However, varying lengths of text from different languages can cause distortions in the layout of the page. Such distortions, called \textit{Internationalization Presentation Failures (IPFs)}, reduce the aesthetics or usability of a website and occur frequently — a recent study reports their occurrence in over 75% of internationalized web pages. Repairing IPFs is challenging for web developers as any kind of style change to one element must be mirrored in stylistically related elements to maintain the aesthetic consistency of the page. To address this problem, I designed an approach, \textit{I}Fix, for automatically repairing IPFs in web pages [4, 5]. I designed a novel clustering technique that identifies groupings of elements that are stylistically similar and adjusts them together in order to maintain the visual consistency of the page. To find repairs, I devised a guided search-based technique that quantifies the amount of distortion in a page by leveraging existing IPF detection techniques and UI change metrics. The empirical evaluation showed
that my approach was able to successfully resolve 98% of the reported IPFs for 23 real-world web pages. In a user study of the repaired web pages, I found that the repairs met with high user approval, with over 70% of the user responses rating the repaired pages as better than the faulty versions.

2.4 \textit{G}Fix: Repair of Mockup Driven Development and Regression Problems

An attractive and visually appealing appearance is important for the success of a website. Presentation failures resulting from Mockup Driven Development and Regression Debugging can negatively impact end users’ perception of the quality of the site and the services it delivers. Debugging such failures is challenging because testers must visually inspect large web pages and analyze complex interactions among the HTML elements of a page. In this work, I designed a novel automated approach for debugging web page user interfaces. My approach uses computer-vision techniques to detect differences between the actual appearance of a web page and its intended appearance, and then analyzes rendering maps of the web page to identify the HTML elements most likely to be responsible for the observed differences [6, 7, 8, 9, 10]. Then I use guided search-based techniques to automatically find repairs for the presentation failures [11]. In the evaluation of the approach on a set of real-world web applications, I found that the approach was able to accurately and quickly identify repairs for the failures.

3 Future Work

Inspired by real-world challenges in repairing presentation failures and the promising results obtained from the different instantiations, in the near future I am interested in continuing this direction of research. I am particularly interested in designing approaches for repairing two types of presentation failures: accessibility issues and problems in responsive web design. Web accessibility is important as it allows inclusion of people with disabilities in the current technology-savvy society and enables them to independently perform work and personal activities, such as shopping, banking, and networking. My key insight is that it is possible to quantify the failures introduced into a page by accessibility issues and use this value as a fitness function to guide a search for a set of HTML/CSS repairs. My idea is that the fitness function can be designed based on existing detectors for accessibility issues, such as AATT and AChecker, and other metrics for measuring the amount of aesthetic similarity of the page before and after repair. The second type of presentation failures I am interested in repairing is problems resulting from responsive web design. Responsive web design is a paradigm that allows developers to design web pages that dynamically adapt their layout to different device sizes. The repair of presentation problems caused by responsive web design would pose new research challenges. For example, analysis to identify which lists of hyperlinks could be grouped into a drop down menu, a refactoring to carry out this change, and a method to quantify the change’s impact. Designing such repairs would be very challenging, as even a seemingly straightforward change, such as changing an element’s size, required the design of graph-based models in \textit{G}Fix and related layout mechanisms to ensure that all other elements that needed to be adjusted would also be included in the change.

In the future, I would like to leverage my knowledge of search-based techniques and use them to convert a given layout of a web page into an energy optimal one while maintaining close aesthetic similarity between the two layouts. The main motivation behind this work is that mobile devices are limited in bandwidth and battery power, making it important for developers to design websites that are energy efficient. My vision in carrying out this work is to first perform a large-scale study for energy profiling of HTML tags and CSS properties to understand their energy saving footprints. A further study would also be performed to analyze energy profiling of different HTML and CSS practices, such as refactoring a table-based HTML layout to a table-less layout and using external versus internal CSS style sheets. Based on these analyses I plan to use search-based techniques to identify the optimal energy efficient refactorings for the page.

In the far future, I would like to retarget my designed framework to areas other than web applications, particularly native mobile apps and apps on IoT (Internet of Things) devices with a Graphical User Interface (GUI), such as a smartwatch, thermostat, and gaming systems. This would pose new research challenges, such as analyzing the characteristics of the presentation failures in these domains, a method to quantify the impact of the failures, and ways to carry out their repair. Although retargeting my repair framework in these new domains would be very challenging, it would allow me to explore new avenues of research.
4 References


