International Journal of Scientific Research in Computer Science and Engineering
Review Paper Vol-1, Issue-3 ISSN: 2320-7639

Cross Browser Testing: A Challenge for Web Testing

Chandraprakash Patidar

Department of Information Technology, Institute of Engineering & Technology DAVV Indore, M.P.-INDIA
chandraprakash_patidar@yahoo.co.in

Available online at www.isroset.org

Received: 05 May 2013 Revised: 20 May 2013 Accepted: 17 June 2013 Published: 30 June 2013

***Abstract*—** Today's world is around internet. To access internet we need a browser. Browsers also have variety and versions. For successful browsing we need compatibility among browsers. Web based applications need to test before deploying. A web application is known as successful when it is executed on all the variety and versions of browsers. For this, web applications should be tested on all types of browsers. This type of testing is known as cross browser testing. Cross browser testing is very important issue as different browsers has its own specification and separate architecture. Today's three most popular browsers are Firefox, Chrome and Internet Explorer. 90% of web testing is performed using these three browsers. Safari also coming close fourth browser. Of course testing only few browsers (Three or Four) is not only the solution of web testing. To cover all web browsers under testing, cross browser testing is very important.

***Keywords*—** Browser, Testing, Web application, Versions.

I. INTRODUCTION

Web applications are increasingly being used for both personal and business activities. Users of such applications might use any web browser to access them, and the application is expected to behave consistently across these different environments. However, web applications often exhibit differences when executed in different browsers, leading to cross-browser inconsistencies (XBIs). XBIs are discrepancies between a web application's appearance, behaviour, or both, when it is run on two different environments. XBIs are not only fairly common, but also notoriously difficult to
Identify and fix. For example, 5328 posts were created and tagged with cross-browser", on stackoverflow.com over the past four years alone. Moreover, nearly 2000 of these posts have been active during the past year [8]. In general, if XBIs are not identified during testing, they can adversely degrade the experience of the users of the web application with the affected browser. In fact, as shown in our evaluation of X-PERT, some XBIs completely prevent users from accessing the functionality offered by the web application, thereby rendering it useless on that particular platform. XBIs are thus a serious concern for companies, which rely on such applications for business or for creating their public brand image. The current practice in industry is to identify XBIs through manual inspection of the web application screens across all the different browsers [2]. Such testing is not only human intensive, but also error-prone.

II. LITERATURE SURVEY

To get a deeper understanding of XBIs, we performed a systematic study of 100 real-world web application[6].Through this study, we were able to establish a classification of XBIs, which further helped us in defining our technique, described in the next section. In particular, we found three
main types of XBIs: structural, content, and behaviour. Structural XBIs: Such XBIs a_ect the structure, or layout, of individual web pages. The web page structure is essentially a particular arrangement of elements, which in case of structural XBIs is erroneous in a particular browser. For instance, the misalignment of one or more web page elements on a given web page, in a particular browser, can constitute a structural XBI.We found that this was the most common category of XBIs, occurring in 57% of the subjects with XBIs. Content XBIs: This kind of XBI is observed in the content of individual components on a web page. Such differences can occur, where the visual appearance of a web page element, or the textual value of an element, are di_erent across two browsers. We further classify these two cases
as visual-content and text-content XBIs. In our study, we found that these XBIs occurred in 30% and 22% of the sites with XBIs respectively. Behavioural XBIs: These type of XBIs involve differences in the behaviour of individual widgets on a web page. An example of such an XBI would be a button that performs a particular action within one browser and a totally different action, or no action at all, in another browser. Another example of behavioural XBI is the presence of an HTML link, which works in one browser but is broken in another one. In our study, such XBIs occurred in 9% of the web applications with XBIs. In summary, behavioural XBIs affect the functionality of individual components, resulting in broken navigation between different screens. Structural and content XBIs, conversely, involve differences in the arrangement or rendering of elements on a particular web page. In the next

section, we describe how our technique detects each of these XBIs.

III. TECHNIQUE OVERVIEW

Algorithm 1 presents an overview of our XBI detection technique. As shown in the algorithm, our technique takes as input the URL of the home page of the web application under test, url, and two browsers considered for the testing, Br1 and Br2. The technique outputs a list of XBIs, X. In this paper, we only summarize the main steps of the algorithm (for all the details, see Reference [6]).

Model Generation via Crawling: The technique starts by crawling the web application, in an identical fashion, in each of the two browsers Br1 and Br2. In this process, it records the observed behaviour as navigation models M1 and M2. The model is captured as a labelled transition system, which represents the top-level structure of the crawled web application. In the model, the states correspond to web application screens, and each transition is labelled with a widget action that leads to a screen navigation. In addition to this navigation model, we also capture the screen image and the DOM structure of the elements on each observed screen. In the algorithm, this step is implemented in function genCrawlModel (line 3).

Behavioral XBI Detection: The navigation models M1 and M2 are checked for equivalence to uncover differences in behavior. To do this, the technique uses the graph isomorphism checking algorithm for rooted labeled directed graphs proposed in [3]. This algorithm is implemented in the different StateGraphs function (line 4), which produces a set of differences (B) and a list PageMatchList of corresponding web-page pairs S1
i ; S2
i between M1 and M2. B contains a set of missing and/or mismatched transitions across pages,

IV. TOOL DESCRIPTION

X-PERT can work with any web application that runs on desktop browsers. Since X-PERT analyses the client-side of such applications, it is agnostic to any server-side technology. X-PERT is written in Python and Java and can run on a variety of desktop operating systems, including Windows, Mac OS X, and Linux. Figure 1 shows a high-level overview of X-PERT, which operates as follows. First, the user invokes the web interface of the tool and interacts with its model generation wizard. This web interface is implemented in Python using the Flask framework (http://flask.pocoo.org) on the server-side, and Twitter bootstrap (http://getbootstrap.com) and jQuery (http://jquery.com) libraries on the client-side. Once the user submits the subject web application's URL and model capture parameters to the wizard, X-PERT uses this information to generate different crawler instances, one for each browser. The generated models are then processed by the model comparison module, which applies our proposed technique to compare these models in a pair-wise fashion. This model comparison module is a key contribution of the X-PERT technique as it compares the different aspects of the web application's execution to uncover the three types of XBIs, which are then gathered, tabulated, and reported to the user. The architecture of X-PERT, shown in Figure 2, consists of the model capture and comparison modules. Both these modules are mainly implemented in Java. Further details of the implementation are discussed below. The Model Capture module uses the Crawljax tool [4], which internally uses the Selenium testing framework (http://seleniumhq.org) to explore the web application in the different web browsers. We extended Crawljax to save the model from its exploration along with the screen shot and DOM structure of each page. The DOM structure is obtained by querying the browser through its Graph Isomorphism Checker.

II. EVALUATION

To assess the usefulness of X-PERT, we ran it on 14 subjects. These subjects are divided in three groups: the rest six subjects were used in prior work, the next four were from our study, and the final four were obtained using an online random URL service (http://www.uroulette.com/).

Our experiments were performed using the latest stable versions of Internet Explorer (v9.0.9) and Mozilla Firefox (v14.0.1). The results of our investigation of X-PERT's effectiveness are shown in Table 1, which lists, for each subject, the XBIs reported in the terms of true (T) and false (F) positives. As shown in the table, X-PERT was effective in finding different kinds of XBIs in the subjects. A deeper investigation of the results [6] revealed that X-PERT's precision and recall are 76% and 95%, respectively, against 18% and 83% for the state-of-the-art tool CrossCheck [5].

V. RELATED WORK

To the best of our knowledge, X-PERT is the _rst tool for comprehensive detection of XBIs. Previous research tools (e.g., [7, 5, 3]) only focused on certain types of issues and had low precision and recall. Developers typically use browser-compatibility tables, such as Quirksmode.org and CanIUse.com, to check their web applications. Some web development tools, such as Adobe Dreamweaver (http://adobe.com/products/dreamweaver.html), provide basic static- analysis based hints to help detect certain issues. However, the issues targeted by reference websites and development tools are limited to features that are known to be missing in a particular browser. Other tools, such as BrowserShots (BrowserShots.org) and Microsoft Expression Web SuperPreview (http://microsoft.com), provide previews of single pages in di_erent browsers, while tools such as CrossBrowserTesting.com and BrowserStack.com allow for browsing web applications in di_erent emulated environments. In both cases, the

comparison of the observed behavior across browsers must still be performed manually.

VI. CONCLUSION

Cross-browser inconsistencies (XBIs) are a serious problem for web developers. Current industrial practice relies on (expensive and error prone) manual inspection to find these issues. Existing research tools, conversely, only target particular aspects of XBIs and can report a significant number of false positives and negatives. To address these limitations, we presented X-PERT, an open source tool for comprehensive XBI detection. Our empirical evaluation shows the effectiveness of X-PERT over the state of the art. This demonstration presents the details of the implementation of XPERT and illustrates how it is fully automated and easy to use through its web interface. In addition, X-PERT generates easy to comprehend and actionable reports for the developer, thus allowing them to address XBIs effectively.

REFERENCES

[1] G. Bradski and A. Kaehler. Learning OpenCV. O'Reilly Media, September 2008.

[2] J. Lewis. Techniques for mobile and responsive cross - browser testing: An envato case study. http://webuild.envato.com 2013.

[3] A. Mesbah and M. R. Prasad. Automated Cross-browser Compatibility Testing. In Proceeding of the 33rd International Conference on Software Engineering (ICSE), pages 561{570. ACM, May 2011.

[4] A. Mesbah, A. van Deursen, and S. Lenselink. Crawling Ajax-based Web Applications through Dynamic Analysis of User Interface State Changes. ACM Transactions on the Web, 6(1):3:1{3:30, March 2012.

[5] S. Roy Choudhary, M. R. Prasad, and A. Orso. CrossCheck: Combining Crawling and Di_erencing to Better Detect Cross-browser Incompatibilities in Web Applications. In Proceedings of the IEEE Fifth International Conference on Software Testing, Veri_cation, and Validation (ICST), pages 171{180. IEEE, April 2012.

[6] S. Roy Choudhary, M. R. Prasad, and A. Orso. X-PERT: Accurate Identi_cation of Cross-browser Issues in Web Applications. In Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, pages 702{711. IEEE Press, 2013.

[7] S. Roy Choudhary, H. Versee, and A. Orso. WebDi_:Automated Identi_cation of Cross-browser Issues in WebApplications. In Proceeding of the 2010 IEEE InternationalConference on Software Maintenance (ICSM), pages 1-10.IEEE, September 2010.

[8] Stackoverow. Posts for cross-browser issues. http://data.stackexchange.com May 2014.

