Using Extracted Attributes to Compare and Look for Cross-Browser Discrepancies

[*]Dr. Kulvinder, [#]Mr. Mohd. Aslam

** Assistant Professor (Computer Science), Tantia University, Sri Ganganagar(Raj.), India*

[#] Mr. Mohd. Aslam (Scholar, Computer Science), University, Sri Ganganagar(Raj.), India

Abstract

The differences between different web browsers are made worse by the development of web technology and the adoption of web apps. These inconsistencies heighten cross-browser incompatibilities, which result in a particular web application having a different appearance on several browsers. Cross-Browser Inconsistencies (XBIs) can either be an acceptable difference or they can completely block users from accessing some of the functionality of an online service. In order to establish consistency, a web application's testing procedure must be carried out thoroughly across several browsers. To identify such problems, it takes a lot of manual labour, and the tools and procedures that are now available offer little assistance in addressing the problems' root causes. In this research, we suggest a method for automatically identifying cross-browser problems.

Currently, client-server websites or web systems have grown into web applications. The server side components are called when a client sends a request to the server through a web browser. These communications cause the server to receive requests, and the server responds by updating the active web page, which is written in HTML (Hyper-Text Markup Language) or XML (extensible Mark-up Language), as well as other relevant resources, such as style information in CSS (Cascading Style Sheets), client-side code (for example, JavaScript), images, and so on. These resources are then employed in the calculation and rendering of an updated web page in the web browser. Ads and generated content (such time, date, etc.) that varies across queries are two common examples of changeable elements in web applications. If these items are not disregarded, the approach can treat them as differences between browsers, leading to false positive results. Therefore, the method necessitates identifying and excluding such components during comparison. A web browser is a piece of software used for finding, viewing, and navigating online information sources. A web page, image, or video is recognized as an information resource by a Uniform Resource Identifier (URL). When the browser requests information, the web server responds. The data is delivered to the web server, which then displays it on the computer. The main issues in using a web application with various web browsers have to do with browser inconsistency. Web apps are also widely employed for all tasks in all professional fields. Cross-Browser Inconsistency is a variance in how elements or content of a web-based application are arranged across various browsers. Cross-Browser Inconsistencies (XBIs) are introduced when a user runs a web application on different browsers since certain web applications display distinct behaviours. When a web

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

application is executed in two separate settings, XBIs show variations in the appearance, functionality, or both. The user experience of a web application may suffer if cross-browser inconsistencies are not properly checked during the testing phase. Therefore, identifying the cross-browser discrepancies is crucial and poses a severe problem for businesses that depend on such programmes. The number of web browser versions released has increased as a result of quickly evolving technology, and browsers are the primary interfaces for delivering/accessing information with a single click. It appears that websites are created for just one browser rather than several. The developer may not be aware of problems that are shown by testing on other browsers. As a result, we carried out a methodical investigation on numerous real-world online apps. This research helps us define our method by enabling the classification of XBIs. There are three main types of XBIs that we discovered: structural, content, and behavioural.

(i) Structural XBIs: These XBIs have an impact on the arrangement or structure of specific web pages. The basic structure of a web page is a specific arrangement of elements, which, in the case of structural XBIs, is incorrect in a specific browser. An example of a structural XBI is the misalignment of one or more elements on a given web page, in a particular browser.

(ii) Content XBIs: These XBIs investigate the content of specific web page components. These variations can occur when an element's textual value or graphical appearance on a web page differs between two browsers. We further divide this kind of inconsistencies into XBIs with visual and text content.

(iii) Behavioural XBIs: These XBIs involve variations in how specific widgets behave on a web page. A button that executes one action in one browser and a completely different action, or no action at all, in another browser is an illustration of an XBI. Additionally, web applications on the internet have subtly emerged as a crucial corporate medium. The failure or poor performance of the business could be caused by software errors in web apps. Making web apps more powerful has been the focus of the majority of the development. However, little is done to ensure the quality Key quality attributes for web applications include reliability, availability, interoperability and security apart from ensuring the functional & usability aspects.

You should leave the technical and challenging task of web browser compatibility testing to your web developer. The issue is that if your website is not compatible with the wide range of available browsers, it will negatively affect the reputation of your company. Techniques that are ideally suited for particular categories of XBIs have been proposed in recent work on recognizing XBIs and focus only on specific features of how a web application is executed. In contrast to CrossT, which employs graph isomorphism in addition to text comparison to find XBIs, the WebDiff programme uses computer vision to identify XBIs. The XBI detection issue can only be partially and erroneously solved by these tools. The XBI detection issue can only be partially and erroneously solved by these tools.

We suggested a method that orchestrates a wide range of comparison approaches to apply each one to the category of XBIs that it is most suited to identify in order to address the

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

shortcomings of existing methods. Our method for XBI detection is a standardized, widely used, and computerized method. The main contributions of this work are:

- A new method and tool for detecting both visual and structural XBIs in web applications;
- A novel, potent method to detect visual XBIs;
- An evaluation of this method on a number of real-world web applications that demonstrates its efficacy in detecting various XBIs.

The remainder of the essay is structured as follows. Introduction to cross browser inconsistency and web applications are found in Section I. Related work done in the domain of cross browser inconsistency of a web application is found in Section II. Finally, the problem specification for our study effort is found in Section III. Section V contains the application area of our study work, Section VI contains predicted results, and Section IV illustrates our proposed technique to discover cross-browser inconsistency with flow chart and Section VII concludes research work with future directions.

CONNECTED STUDY

The needs for testing infrastructure are no longer static due to the growth of various browser versions and release updates. Cross-browser compatibility has become a significant difficulty for software testers due to the numerous smart gadgets that are flooding the market every day.

It has been noted that TCS emphasizes the success of the Cross Browser Testing Tool. It provides a computerized solution with a set of preconfigured devices and a test environment that makes it easy to test quickly across many operating systems and browsers. It establishes connections with actual hardware to test web-based mobile applications and guarantees accurate cross-browser and cross-device testing. It addresses three significant testing domains: UI validation across browsers, Functional testing ensures that functionality is accurate, while RWD testing addresses page consistency issues while assuring the best viewing and user interaction across a variety of devices. Additionally, they provide a solution that enables cross-browser compatibility by using efficient layout comparison, functional testing, responsive web design, broken link validation, portal-based management, and obtains up to a 60% reduction in test effort through the use of dynamic script development. It offers centralized test administration and an improved UI scanning mechanism. With automatic PDF evaluation and parallel test implementation across several browser versions, about 50% of time is saved. Additionally, the specifics of a widely used methodology for website cross compatibility testing have been provided. It addresses the technical challenges of a website and how identifying cross-browser inconsistency is necessary due to variations in browsers, operating systems, and devices Additionally, the requirements that a website must meet before going live worldwide and some cross-browser automated testing tools that help with website testing on a variety of browsers, operating systems, and devices have been suggested. These tools also meet the technical specifications for guaranteeing website quality. Then, a variety of tools have been contrasted based on their speed, pricing structure, interfaces, delays, scroll bars, and other aspects. Accordingly, the web performance testing for web site

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

functionality on different web browsers, operating systems and different hardware platforms is checked for software, hardware memory leakage errors. The relative analysis of cross-browser compatibility as a design issue in various websites has then been developed utilizing an online application built on the.NET Framework. It offers numerous design and development issues for many types of websites, including governmental, educational, commercial, social networking, and job portal websites. The findings of evaluating five major types of websites reveal that educational and social networking websites have the least cross-browser compatibility, whereas job portals, commercial websites, and government websites adhere strictly to the W3C's recommendations for website design.

A separation technique that isolates the performance of one application from that of another has also been published, in an effort to turn the browser into a secure environment for running programmes. It demonstrates how to safely divide programmes using OS processes within the browser in a way that is both effective and backwards compatible with already-existing websites. Additionally, it detects whether a web page's content is more active or richer active, as well as browser issues like failure separation, concurrency, and memory management. These have demonstrated that existing web browsers offer uncertain environments for running apps through assessments of both site content and browser behaviour.

As a result, failure isolation, concurrency, and memory management all experience significant issues. These have demonstrated that employing OS processes, browser-based programmes may be safely segregated from one another. Processes stop unauthorized communication between browser applications, and they are well-organized in terms of performance and memory overhead as compared to other browser processes. Then, a quantitative categorization of browser vulnerabilities has been presented in order to project the quantities of vulnerabilities for more effectively allocating test and improvement resources. Internet Explorer, Firefox, and Mozilla's vulnerability discovery data have been evaluated, fixed to a vulnerability discovery model, and the integrity of fit has been statistically tested. Additionally, it categorizes vulnerabilities according to source, impact, severity, and reason. Classification of Vulnerability such as Input Validation Error (includes boundary condition error, buffer flood), Access Validation Error ,Exceptional situation Error ,Environmental Error, Configuration Error, race Condition Error, Design Error.

Later, the challenge of automating cross-browser testing of online applications as a functional consistency check of their behaviour across several web browsers has been raised. In this method, the given web application is automatically examined in a variety of browser contexts to capture the behaviour as a finite-state machine. The created models are then compared for equivalence on a pair-by-pair basis, and any inconsistencies are then made clear. There are two steps in this overall strategy. The first phase entails autonomously crawling the specified online application across various browser environments, capturing and storing the observed behaviour across each browser as a unique state machine navigation model. The crawling is done in an identical fashion under each browser to replicate precisely the same set of user interaction sequences with the web application, under each environment. The second step

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

consists of formally comparing the generated models for similarity on a pair wise-basis and revealing any experimental discrepancies.

Additionally, a programme that automatically detects XBIs in web applications without the developer's involvement has been made available. Any web application that operates on desktop browsers can be used with this tool. This model takes a screenshot of the screen and uses the graph isomorphism checking method to compare the graph produced by the crawler. Additionally, it locates several kinds of irregularities in a web application. Additionally, it produces reports for the developer that are simple to read and useful, enabling them to handle XBIs more effectively.

The work load produced by Google's Chrome browser on a heterogeneous multi-processing (HMP) platform seen in many smart phones has thus been studied at the thread level. The detailed analysis of the web browser's thread burden, particularly the rendering engine it examines, and discussion of power-saving options in connection to Android power management policies. Additionally, it places a focus on how to control web browser workload on HMP systems while also looking for potential power savings based on the interpretation.

Its new functionality focuses on a function call made by the web browser and the real thread workloads. Additionally, it offers data that can be applied to power management. Then, various tools enabling parallel execution of a range of automated tests using several remote test environments with different web browsers have been recommended. It also presents a tool for automated testing of web applications based on the Selenium RC framework.

DEFINITION OF A PROBLEM

Any web browser can view a website with the same appearance thanks to high-quality web design. As a result, any web browser must be able to access a high-quality website in all of its capabilities. Every webpage consists of a variety of distinct components, each of which influences how well it functions in certain situations. The browser compatibility aspect of a website is similarly impacted by different webpage components, either directly or indirectly, as are other performance assessment metrics. Additionally, the compatibility problem is brought on by various technologies. As a result, websites must undergo extensive testing throughout the design phase to ensure that they are compatible with various viewing contexts. Section I discussed about the parameters that can affect the cross browser inconsistency of a web application.

RECOMMENDED SOLUTION

Description

We suggest a model to detect cross-browser inconsistencies (XBI) in order to find them. Figure 1 shows a high-level perspective of our suggested XBI detection method, which requires as inputs the URL of the web application under test's home page, URL, and the two testing browsers, Browser1 and Browser2. The result is a list of the found discrepancies. Our suggested model contrasts properties retrieved from a produced graph by a crawler.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

(i) Web crawler

An automated programme or script known as a web crawler carefully examines online pages to create an index of the information it is programmed to seek for. Crawling the web is also known as spidering. The web crawler "WebSPHINX (Website-Specific Processors for HTML Information extraction)" written in Java was our suggested method of use. It crawls many webpages and generates a corresponding graph. Websphinx.zip contains the source code for this open source web crawler.

(ii) Attribute Extractor

The foundation of attribute extractor is as follows: Since attribute terms repeat across numerous graphs for a web application, they are more likely to appear in a graph generated by a web crawler than other terms. To capture the qualities, we try to take use of this redundancy.

Thus, choosing attributes would be easiest if the graphs' most prevalent phrases were used. This approach, however, has a flaw. This approach only provides frequent qualities and is likely to ignore rare attributes that appear in a small number of graphs. To overcome the first problem, we propose a two stage method. In the first stage, we cluster the all the words found in the graphs such that all the words close to an attribute are grouped together in a single cluster. This results in word clusters of different sizes. In the second stage, we extract an attribute from each cluster.

(iii) Comparator

To find text-content XBIs, this module performs textual analysis on the related elements. It compares screen images of the corresponding web page elements in order to find image-content XBIs. The Layout Analysis component examines the structure of the page that was scraped by the crawler in order to produce alignment graphs, which show how web page elements are positioned in relation to one another. Comparisons can be done in two ways: pair wise, where two graph attributes are compared, or three ways, where three graph attributes are compared.

(iv) Classifier

This module categorizes the several types of inconsistencies found in web applications, such as structural, content, and behavioural inconsistencies.

(v) Reporting Tool

This module creates an HTML report that tabulates the collection of discovered XBIs.

APPLICATION

E-commerce websites, commercial websites, educational websites, government websites, news portals, and social networking websites are application areas for detecting cross-browser inconsistency. There is a requirement that a web application function similarly when run on a variety of different browsers for the same reason.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

ANTICIPATED RESULTS

It has been found that when a web application is run on several browsers, our suggested model should be able to detect three different kinds of discrepancies, if any. Additionally, this suggested model produces a summary of errors. Due to the three-way comparator used in this technique, which compares three crawler-generated graphs at once. Therefore, compared to other methods, discovering XBIs can be a quick technique. These are shown as follows:

- We discovered that structural XBIs, which affect 57 percent of people with XBIs, are the most prevalent type of XBI.
- We establish that these content XBIs happened in 30 percent and 22 percent, respectively, of the sites hosting XBIs.
- We find that 9 percent of the web applications having XBIs had behavioural XBIs.

We can therefore draw the conclusion that the behavioural XBIs have an impact on the functionality of specific components, leading to problematic screen switching. Contrarily, structural and content XBIs refer to variations in the organization or presentation of elements on a certain web page.

CONCLUSION

Web developers have a serious problem with XBIs. Existing research techniques can produce a high number of false positives and false negatives since they only focus on a specific feature of XBIs.

We presented our suggested methodology for XBI detection to address these constraints.

This paper provides an overview of the suggested technique, examples of its use, and anticipated outcomes to achieve this aim. Additionally, it generates simple-to-understand information for the developer, enabling them to handle XBIs more successfully. This is a significant challenge because the programme will need to look different on the two platforms even though it should provide the same functionality, if not somewhat equivalent capabilities.

REFERENCES

[1] C.P.Patidar and Meena Sharma ,"An automated approach for cross browser inconsistency(XBI) detection", Ninth annual ACM India conference organized by ACM India, Oct 21-23,2016.

[2] Nepal Barskar, C.P.Patidar and Meena Sharma, "Analysis and Identification of Cross Browser Inconsistency Issues in Web Application using Automation Testing", International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555Vol.6, No3, MayJune 2016.

[3] Nepal Barskar and C.P. Patidar, "A Survey on Cross Browser Inconsistencies in Web Application", International Journal of Computer Applications (0975 – 8887) Volume 137 – No.4, March 2016.

[4] "WebTesting", mindlance.com, testing@mindlance.com.

[5] Ochin and Jugnu Gaur, "Cross Browser Incompatibility: Reasons and Solutions", International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.3, July 2011.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

[6] Shauvik Roy Choudhary, Husain Versee and Alessandro Orso, "WEBDIFF: Automated Identification of Crossbrowser Issues in Web Applications", 26th IEEE International Conference on Software Maintenance in Timisoara Romania, 978-1-4244-8628-1/10, 2010.

[7] http://www.tcs.com/assurance, 2016.

[8] Sanjay Dahiya1, Ved Parkash1 and T.R. Mudgal2, "Comprehensive Approach for Cross Compatibility Testing of Website ", National Workshop-Cum-Conference on Recent Trends in Mathematics and Computing (RTMC) ,Proceedings published in International Journal of Computer Applications (IJCA) ,2011 .

[9] Jatinder Manhas," Comparative Study of Cross Browser Compatibility as Design Issue in Various Websites" , BIJIT - BVICAM's International Journal of Information Technology Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM), New Delhi (INDIA), NOV 2014.

[10] Charles Reis, Brian Bershad, Steven D. Gribble and Henry M. Levy, "Using Processes to Improve the Reliability of Browser-based Applications", University of Washington Technical Report UW-CSE, DEC 2007.

[11] Sung-Whan Woo, Omar H. Alhazmi and Yashwant K. Malaiya,"AN ANALYSIS OF THE VULNERABILITY DISCOVERY PROCESS IN WEB BROWSERS", proceeding of the 10th IASTED International Conference Software engineering and Applications, Dallas,TX,USA,ISBN: 0-88986-642-2 / CD: 0-88986-599- X, NOVEMBER 13-15, 2006,

[12] Ali Mesbah and Mukul R. Prasad," Automated CrossBrowser Compatibility Testing", ICSE '11,Waikiki, Honolulu, HI, USA ,ACM 978-1-4503-0445-0/11/05, May 21–28, 2011 .

[13] Shauvik Roy Choudhary, Mukul R. Prasad and Alessandro Orso," X-PERT: A Web Application Testing Tool for Cross-Browser Inconsistency Detection", ISSTA'14,San Jose, CA, USA,Copyright 2014 ACM 978-1-4503-2645- 2/14/07,July 21–25, 2014.

[14] Nadja Peters1, Sangyoung Park1, Samarjit Chakraborty1, Benedikt Meurer2, Hannes Payer2 and Daniel Clifford2, "Web Browser Workload Characterization for Power Management on HMP Platforms", CODES/ISSS '16 Pittsburgh, PA, USA, ACM, and ISBN: 978-1-4503-4483- 8/16/10, October 01-07 2016. [15] Shauvik Roy Choudhary, Mukul R. Prasad and Alessandro Orso, "CROSSCHECK: Combining Crawling and Differencing to Better Detect Cross-browser Incompatibilities in Web Applications", 2012.

[16] Shauvik Roy Choudhary,"Detecting Cross-browser Issues in Web Applications", ICSE '11, Waikiki, Honolulu, HI, USA, ACM 978-1-4503-0445-0/11/05, May 21–28, 2011.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731