IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-7, July 2018 E-ISSN: 2347-2693

Implementation of Visual XBI Detector by Comparing RGB Index and Histograms

Neha Verma[1*], C.P. Patidar[2]

[1*]Computer Engineering, IET, DAVV, Indore, India
[2]Information Technology, IET, DAVV, Indore, India

[*]*Corresponding Author: verma.neha1804@gmail.com, Tel.: +91-9713048596*

Available online at: www.ijcseonline.org

Accepted: 20/Jul/2018, Published: 31/Jul/2018

***Abstract*—** There are tons of browsers in the market which are being used to surf internet. Each browser has its own DOM parser, rendering engine which has its own sense of understanding the content of a webpage pulled from the web server. It parses the html tags, images and displays the same accordingly on the screen. This is the main cause of visual xbi's in different browsers. Companies are spending a big lump of money only for the look and feel of the sites. Numerous front-end technologies are being developed with time to enhance the web development as independent of different browsers. The main intent of this research is to identify the visual difference in how the images are being displayed by different browsers as there are different rendering engines in the market. Studying the RBG values of an image at pixel level, generating histograms and exploiting the DOM structure to extract co-ordinates of an images helps in studying that how the images in a webpage are rendered by different browsers.

***Keywords*—** Browser,CrossBrowserInconsistency,Reliability,Webapplication,DOM,RGB

I. INTRODUCTION

Cross-Browser Inconsistencies (XBIs) are inconsistencies between a web application's appearances, conduct, or both, at the point when the application is keep running on two distinct conditions. Because of the expanding prevalence of web applications, and the quantity of programs and stages on which such applications can be executed, XBIs are a genuine worry for associations that create electronic programming.

Visual XBIs speak to contrasts in the visual appearance of individual page components (e.g., contrasts in the styling of content or back-ground of a component crosswise over different programs relate to visual-content XBIs). Such blunders must be seen in the picture portrayal of the elements [1]. Hence, the method measures the separation between the shading histograms of the component's screen picture. A web program is a product application for recovering, exhibiting, and navigating data assets on the World Wide Web. By a Uniform Resource Identifier (URL), a website page, picture and video a data asset is perceived. The program gets in contact with the web server and requirements for data. The web server gets the data and showcases it on the PC. The significant issues partners with utilizing the web application through various web programs are connected with web program irregularity. Additionally, web applications are being utilized by numerous for all exercises in each field of work. Some variety in course of action of components or substance of an online application on various programs is known as Cross-Browser Inconsistency [2] .When a client execute a web application on numerous programs, at that point some web application display diverse practices and in this manner presents Cross-Browser Inconsistencies (XBIs). XBIs display contrasts between a web application's appearances, conduct, or both, when it is executed on two unique conditions. In the event that cross program inconsistencies are not being accurately tried amid the testing stage, at that point it can contrarily influence the experience of the client of web application [3].

The rendering motor will begin parsing the HTML archive and turn the labels to DOM hubs in a tree called the "substance tree". It will parse the style information, both in outside CSS documents and in style components. The styling data together with visual directions in the HTML will be utilized to make another tree - the render tree.

The render tree contains square shapes with visual traits like shading and measurements. The square shapes are organized appropriately to be shown on the screen.

After the development of the render tree it experiences a "format" process. This implies giving every hub the correct directions where it ought to show up on the screen. The following stage is painting - the render tree will be navigated and every hub will be painted utilizing the UI backend layer.

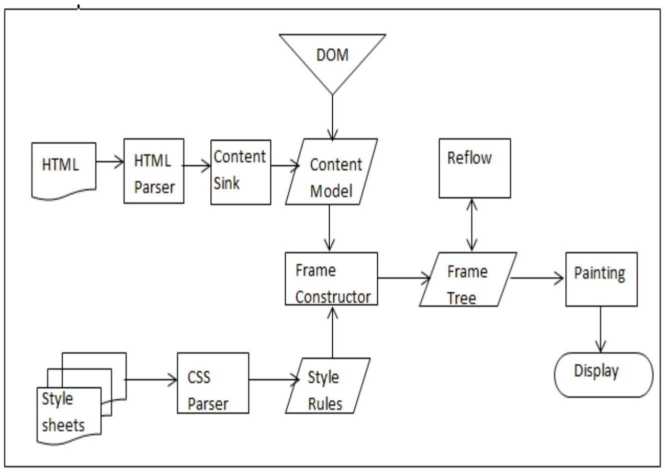


Fig.1 Flow of web parsing

II. CROSS-BROWSER INCONSISTENCIES

A critical issue in this area is to recognize inconsistencies emerging because of the distinction in the application's conduct when it keeps running on two unique stages. On account of web applications, these inconsistencies can be seen when the web application is run on various web programs. It brings about Cross-Browser Incompatibilities (XBIs) which are errors in a web application's appearance, usefulness, or both, when the application is keep running on two diverse web program situations.
As per the work, XBIs can be categorized as three primary XBIs: Behavior, Structure and Content.

• **Behavior XBI**: This XBI demonstrates the distinction in the conduct in same component of a website page in various programs Case for such XBI is if a catch playing out some activity like submitting or continuing to next page yet a similar catch performs diverse activity when keep running on any another program[4].

• **Structure XBI**: Such XBIs influence the structure, or format, of individual pages. The site page structure is basically a specific course of action of components, which if there should arise an occurrence of auxiliary XBIs is mistaken in a specific program. This current XBI's alludes to the distinction in the format of the page. For instance, two catches in the pages are organized on a level plane (departed to appropriate) in one program, yet vertically in another program.

• **Content XBI:** This sort of XBI is seen in the substance of individual segments on a site page. Such contrasts can happen, where the visual appearance of a website page component, or the printed estimation of a component, are distinctive crosswise over two programs. These can be additionally delegated visual-substance and content substance XBIs. It alludes to the distinction in the substance of individual segments of the website page. It can be additionally named content substance XBI and visual substance XBI. The previous includes the distinction in the content estimation of a component, though the last alludes to the distinction in the visual part of a solitary component (e.g., page title has shadow in Firefox and no shadow in Internet Explorer).
The XPERT device was intended to distinguish the three principle kind of inconsistencies. It takes information at that point contrast it with various programs and distinguishes which kind of inconsistency is present. For the most part XPERT focused on content substance XBI to distinguish visual inconsistencies they utilized OpenCV toolbox.
In this paper we will focus around visual inconsistencies and will figure out how to distinguish the visual inconsistencies.

III. RELATED WORK

To the best of our insight, X-PERT is the principal device for exhaustive identification of XBIs. Past research devices just focused on specific kinds of issues and had low exactness and review [5]. Engineers commonly utilize program similarity tables, for example, Quirksmode.org and CanIUse.com, to check their web applications. Some web improvement devices, for example, Adobe Dreamweaver (http:// adobe.com/items/dreamweaver.html), give essential static examination based insights to help distinguish certain issues. However, the issues focused by reference sites and advancement devices are constrained to highlights that are known to be absent in a specific program. Different instruments, for example, BrowserShots (BrowserShots.org) and Microsoft Expression Web SuperPreview (http://microsoft.com)[6], give pre- perspectives of single pages in various programs, while devices such as CrossBrowserTesting.com and BrowserStack.com consider perusing web applications in various copied environments. In the two cases, the correlation of the watched conduct crosswise over programs should at present be performed physically.

IV. TECHNIQUE OVERVIEW

We implemented our approach in a crawler called CrawlJax which is implemented in Java and accepts a web page and extracts images from that web page. CRAWLJAX acts as a driver and, by triggering actions on web page elements, is able to explore a finite state space of the web application and save the model as a state-graph representation. CrawlJax extracts an image of the entire webpage and geometrical information about the page elements.
First of all the data store, it saves the images of web page and DOM structure of web page and rest of the information too. Then it creates the DOM tree of that web page, after that

image extractor extracts all the images that are available on that web page. In this we used the selenium to automate the process. We are generating histogram of a particular image that is later compared to identify that visual consistency occurs or not.

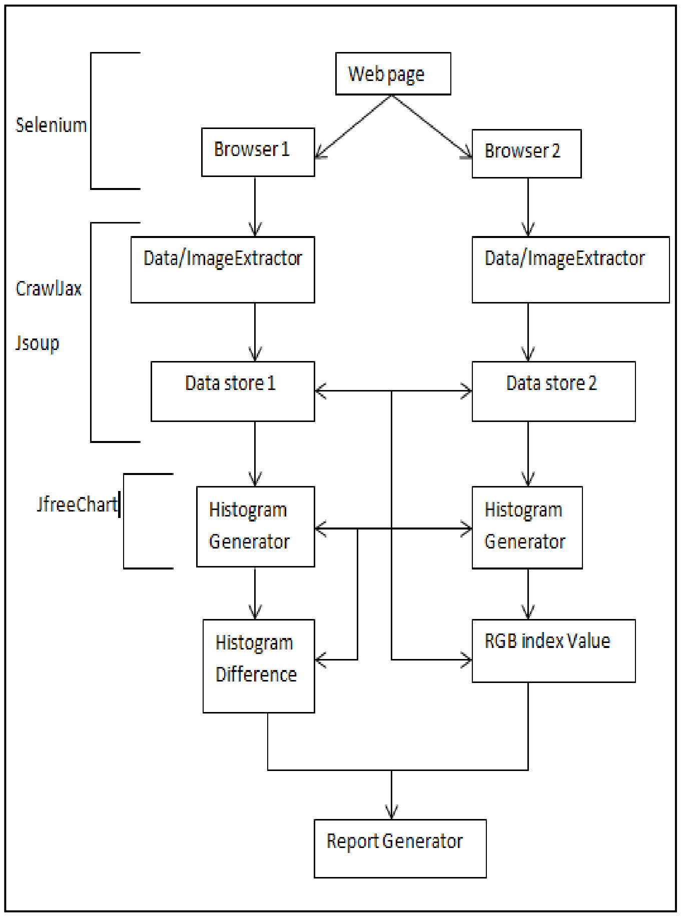


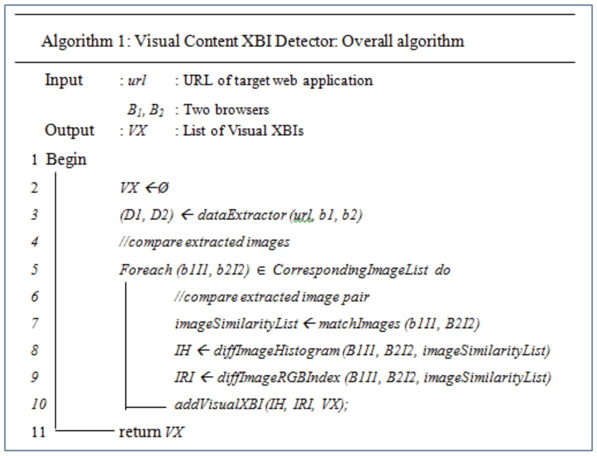
Fig.2- Flow diagram of VXBI Detector

After that RGB index of an image is inspected the colour of an image is expressed as an RGB triplet (r, g, b), each component of which can vary from zero to a defined maximum value.

Later, image comparator compares the image which is in the form of histogram and DOM is compared. This tool is generally working in two different parts, first one is data extractor which extracts all the details regarding practical which we are performing and second is data analyser which analyse all the data.

Algorithm 1 presents an overview of our VXBI 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, B1 and B2. The technique outputs the visual difference in images VX.

The technique starts by crawling the web application, in an identical fashion, in each of the two browsers B1 and B2. In this process, it records the observed behavior as data extractor D1 and D2. The model represents the top-level structure of the crawled web application. In the model, the states correspond to web application screens. In addition to this data extractor, we capture the screen image and the DOM structure of the elements on each observed screen.



```
Algorithm 1: Visual Content XBI Detector: Overall algorithm
Input   : url      : URL of target web application
          B1, B2   : Two browsers
Output  : VX       : List of Visual XBIs
1  Begin
2      VX ←Ø
3      (D1, D2) ← dataExtractor (url, b1, b2)
4      //compare extracted images
5      Foreach (b1I1, b2I2) ∈ CorrespondingImageList do
6          //compare extracted image pair
7          imageSimilarityList ← matchImages (b1I1, B2I2)
8          IH ← diffImageHistogram (B1I1, B2I2, imageSimilarityList)
9          IRI ← diffImageRGBIndex (B1I1, B2I2, imageSimilarityList)
10         addVisualXBI (IH, IRI, VX);
11 return VX
```

Fig.4-Algorithm to detect VXBI

The data extractor D1 and D2 are checked for equivalence to uncover differences in vision. To do this, the technique uses the CrawlJax, which produces a set of differences and a list ImageSimilarityList of corresponding web-page pair b1I1 and b2I2 from pages b1 and b2. ImageSimilarityList contains the mapping between corresponding screens across browsers. This is implemented by CrawlJax, which compares url b1 and b2 and extracts the set of available image on that page. These XBIs represent differences in the visual appearance of individual page elements (e.g., differences in the styling of text or back-ground of an element across different browsers correspond to visual-content XBIs). Such errors can only be observed in the image representation of the elements. Hence, the technique crawl the web page and save all the images available on that page for two different browsers. After extraction the algorithm generates the histogram of two corresponding images of two different browsers i.e. chrome and explorer. Hence, the technique measures the distance between the colour histograms of the image and compares the RGB index of images that gives the difference in images.

V. TOOL DESCRIPTION AND IMPLEMENTATION

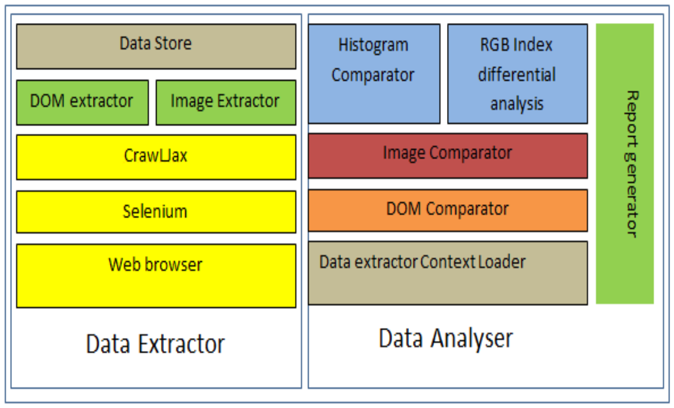


Fig.3 Architecture of VXBI Detector

➢ **Crawl Jax –** CrawlJax is an open source java tool for automatically crawling and testing modern web applications. CrawlJax can explore any javascript-based ajax web application through an even-driven dynamic crawling engine. It automatically creates a state-flow graph of the dynamic DOM states and the event-based transitions between them.

➢ **Selenium –** Selenium is a versatile programming testing system for web applications [7]. Selenium gives a playback (some time ago likewise recording) apparatus for creating tests without the need to take in a test scripting dialect (selenium ide). The tests would then be able to keep running against most current web programs.

➢ **Image Extractor –** This module parses the state data saved using ***JSoup*** library and creates a list of images present in a state. Further it extracts all images from the state and save all the images to *CrawlerOuput* directory state wise.

➢ **Data Store –** This component is used to persist and load the pages from the file system as XML files. These components are implemented using the Object serialization support in Java and are essential for the model capture to compare components [8]. For instance, model capture might be run to collect images from different browsers.

➢ **Histogram Comparator –** *HistogramGenerator* Module generates histogram for all the images saved by *ImageExtractor* module using ***JFreeChart*** java open source library. All the histograms are stored in *Histograms* directory state wise. It compares histograms of corresponding images of a specific state using ***JFreeChart*** library.

➢ **RGB Index Differential Analysis –** RGB index differential analysis constructs all the colours from the combination of the Red, Green and Blue colours. This module generates the RGB index of two same images which are extracted from two different browsers. The algorithm gets red, green and blue component's value for each pixel. It calculates the difference in two corresponding pixel of three components. The difference for all the pixels (N – total pixels) is added and divided by below three values –

N – To obtain average difference per pixel
3 – To obtain average difference per colour component
255 – To obtain value between 0.0-1.0

$$\frac{\lim_{0\to N} [(RedA- RedB)+(GrnA-GrnB)+(BLuA-BluB)] * 100}{3 * N * 255}$$

➢ **Image Comparator –** It compares screen images of the corresponding elements on the webpage. Specifically, this image comparison measures the distance between their colour histograms to detect image-content XBIs.

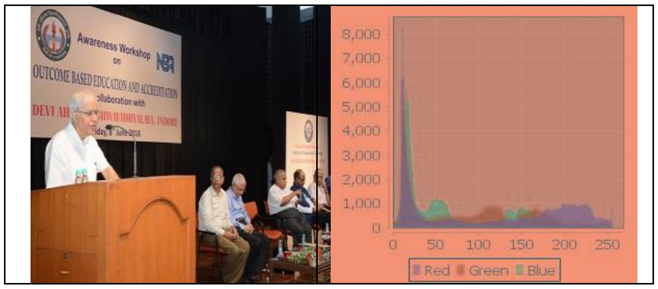


Fig.5–Histogram for DSC_workshop080618.jpg http://www.dauniv.ac.in/

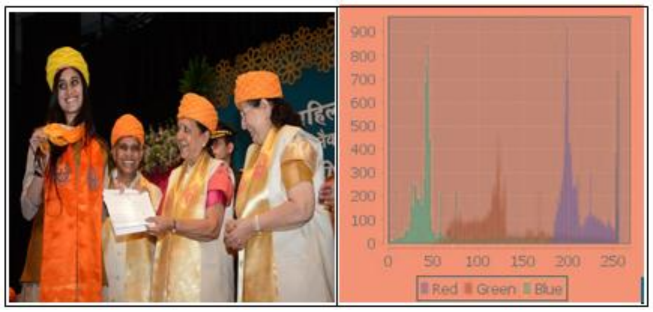


Fig.6–Histogram for DSCConvocation.jpg http://www.dauniv.ac.in/

➢ ***Dom Comparator*** – It compares DOM tree of two different web pages. This module matches comparing components on renderings of a site page over the different browsers by computing the match file metric.

VI. EVALUATION

To assess the usefulness of VXBI detector, website (http://www.dauniv.ac.in/) was tested. The process is divided in four groups: the first shows the states of the link, second shows the RGB difference of these images, third shows the histogram difference and fourth shows the coordinate difference.

Our experiments were performed using the latest stable versions of Internet Explorer (v9.0.9) and Google chrome [v14.0.1]. the results of our investigation of VXBI's effectiveness are shown in Fig-4, which lists, for each input the VXBI reported difference in percentage and the term *size mismatch*. As shown in the table, VXBI is effective in finding the visual difference in different images.

VII. RESULT

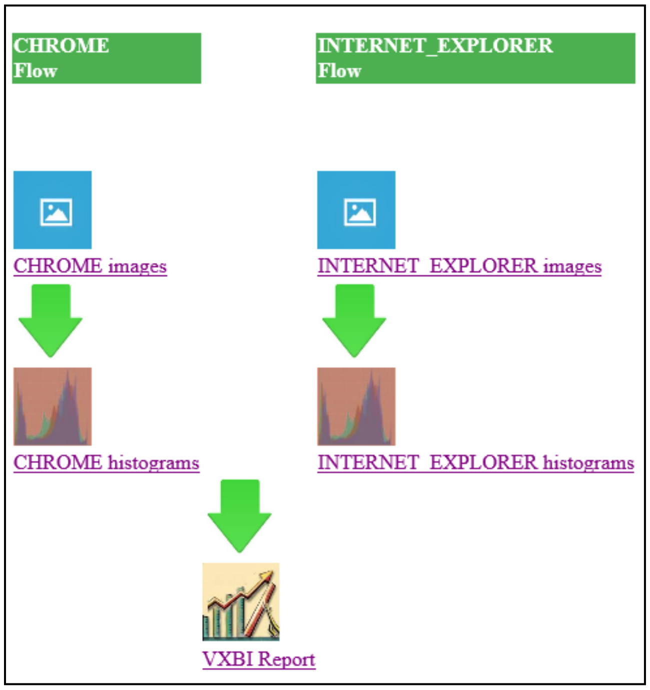


Fig.7 -VXBI tool Report

State Name	RGB Difference	Histogram Difference	Co-ordinate difference
Index.jpg			Size Mismatch
Index_small.jpg	49.65071568627451	49.65071568627451	
State11.jpg			Size Mismatch
State11_small.jpg	73.98968954283665	73.98968954283665	
State12.jpg			Size Mismatch
State12_small.jpg	78.57891657951566	78.57891657951566	
State14.jpg			Size Mismatch
State14_small.jpg	54.15485546548659	54.15485546548659	
State14.jpg			Size Mismatch
State18.jpg	69.54895541534545	69.54895541534545	
State18_small.jpg			Size Mismatch
State4.jpg	54.48754511548787	54.48754511548787	
State4_small.jpg			Size Mismatch
State6.jpg	76.45846845151548	76.45846845151548	
State6_small.jpg			Size Mismatch

Fig.8-The VXBI Difference result for states

Result shows the RGB diference which is calculated by given formula(*RGBIndexDifferentialAnalysis*),the histogram difference Which is calculated by comparing two same images from two different browser and the coordinate difference which is calculated by comaparing the coordinates of two same images of different browsers.

State Name	Image Name	RGB Difference	Histogram Difference
Index.html	0.png	0.0	0.0
Index.html	3.png	0.0	0.0
Index.html	50.png	0.0	0.0
Index.html	Convocation310318image1.jpg	0.0	0.0
Index.html	Convocation310318image2.jpg	0.0	0.0
Index.html	ConvocationImage3_310318.jpg	0.0	0.0
Index.html	Council.jpg	0.0	0.0
Index.html	DAVV_Apps2.jpg	0.0	0.0
Index.html	DSC_Wo080618.jpgrkshop	0.0	0.0
Index.html	Facilities.jpg	0.0	0.0
Index.html	Logo.png	0.0	0.0
Index.html	Photo.png	0.0	0.0

Fig.9-The VXBI Difference result for images in states

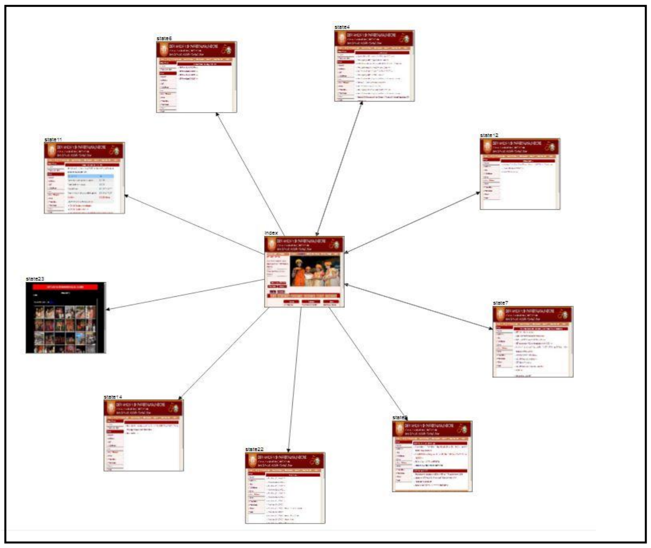


Fig.10-The state graph for http://www.dauniv.ac.in/

This is the state graph of DAVV website which is given as an input to crawler. The crawler saves all the visited states. We can change the number of states to be visited.

Crawl Result

Crawl Time	51 seconds.
Exit Status	Maximum States passed
Number of Status	10
URL's Visited	10
Number of edges	20
Number of crawl paths	17
Failed events	0
Average DOM length	16,651 kB

Fig.11- Result Statistics

The result statistics shows that how much time was taken to crawl the website, It keeps the count of states, It maintains the track of exit status, number of edges, number of crawl path, failed event and average DOM length and the visited URL.

VIII. CONCLUSION

Cross-program inconsistencies (XBIs) are a genuine problem for web designers. Current modern practice depends on (costly and blunder inclined) manual investigation to discover these issues. Existing exploration devices, on the other hand, just target standard particular parts of XBIs and can report a significant number of false positives and negatives. To address these limitations, we displayed this paper on an open source apparatus for comprehensive VXBI recognition. Our exact assessment appears the effectiveness of this device over the cutting edge. This show introduces the subtle elements of the usage of apparatus and outlines how it is completely robotized and simple to use through its web interface. What's more, apparatus generates simple to understand and significant reports for the engineer, hence enabling them to address visual XBIs effectively.

REFERENCES

[1]C.P.Patidar and Meena Sharma ,"An automated approach for cross browser inconsistencies(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, May-June 2016.

[3] A. Mesbah and M. R. Prasad, "Automated cross-browser compatibility testing," in Proceeding of the 33rd International Conference on Software Engineering, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 561–570.

[4] Choudhary, S. R., Prasad, M. R., & Orso, A., "X-PERT: A Web Application Testing Tool for Cross-Browser Inconsistency Detection", 2014.

[5] S. Roy Choudhary, M. R. Prasad, and A. Orso. X-PERT: Accurate Identifcation 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.

[6]S. R. Choudhary, M. R. Prasad, and A. Orso. X-PERT: Accurate Identification
of Cross-Browser Issues in Web Applications. In *Proceedings of* the 35th IEEE and ACM SIGSOFT International Conference on Software
Engineering (ICSE 2013*)*, pages 702–711, May 2013.

[7]S. Mahajan and W. G. Halfond. Root Cause Analysis for HTML Presentation Failures Using Search-based Techniques. In Proceedingsof the 7th International Workshop on Search-Based Software Testing*(SBST)*, Hyderabad, India, June 2014

[8] S. Roy Choudhary and A. Orso, "Webdiff: Automated identification of cross-browser issues in web applications," in ICSM '10: Proceedings of the International Conference on Software Maintenance. IEEE, September 2010.

[9]Sonal Mahajan and William G. J. Halfond. 2015. Detection and Localization of
HTML Presentation Failures Using Computer Vision-Based Techniques. In Proceedings
of the 8th IEEE International Conference on So.ware Testing, Verifcation and Validation (ICST).

[10]Mesbah, A., & Prasad, M. R., "Automated cross-browser compatibility testing". 33rd International Conference on Software Engineering, ACM Proceedings, pp. 561-570, 2011.

[11]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.

[12] 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, May-June 2016.

[13] 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.

[14] Sonal Mahajan, Abdulmajeed Alameer, Phil McMinn, and William G.J. Halfond.
2017. XFix: Automated Tool for Repair of Layout Cross Browser Issues. In Proceedings of the 26th International Symposium on So.ware Testing and Analysis (ISSTA) – Tool Track.