3rd International Conference on Applied Computing & Information Technology
(ACIT 2015)

July 12 – July 16, 2015
Okayama, Japan

Editors:
Kensei Tsuchida
Takaaki Goto

International Association for Computer and Information Science (ACIS)
URL: www.acisinternational.org
Table of Contents

KEYNOTE
Biomedical Text Mining: Experience and Practical Approach
Keun Ho Ryu ................................................................. 1

TRACK 1: SOFTWARE ENGINEERING
Improving Relevancy Filter Methods for Cross-Project Defect Prediction
Kazuya Kawata, Sousuke Amasaki, Tomoyuki Yokogawa ........................................ 3
Automatic Method of Generating a Web Prototype Employing Live Interactive Widgets for Validation of Functional Usability Requirements
Shohei Kamimori, Shinpei Ogata, Kenji Kaijiri ............................................................... 9
Fu-Rin-Ka-Zan: Quantitative Analysis of Developers’ Characteristics based on Project Historical Data
Atsushi Itsuda, Shin Fujivara, Nao Yamasaki, Haruaki Tamada, Hideaki Hata, Masateru Tsunoda ....................... 15
Benchmarking Software Maintenance Based on Working Time
Masateru Tsunoda, Akito Monden, Kenichi Matsumoto, Sawako Ohiwa, Tomoki Oshino .................. 21
Do Learners to Create an Artifact with Good Quality Make a Number of Trials and Errors during the Editing Process?
Takafumi Tanaka, Hiroaki Hashiura, Atsuo Hazeyama, Seiichi Komiya ................................. 29
Minimizing Over Design and Under Design
Yucong Duan ........................................................................ 35
A Requirement Management Education Support Tool for Requirement Elicitation Process of REBOK
Tetsuro Kakeshita, Satoshi Yamashita ................................................................. 41

TRACK 2: GAME INFORMATICS
Satogaeri, Hebi and Suraromu are NP-Complete
Shohei Kanehiro, Yasuhiko Takenaga ................................................................. 47
Adaptive Fighting Game Computer Player by Switching Multiple Rule-based Controllers
Naoyuki Sato, Sila Temsiririkkul, Shogo Sone, Kokolo Ikeda ........................................ 53
An Approach to Quantifying Pokemon’s Entertainment Impact with focus on Battle
Panumate Chetprayoon, Shuo Xiong, Hiroyuki Iida ....................................................... 61
A Decision Making Method Based on Society of Mind Theory in Multi-player Imperfect Information Games
Mitsuo Wakatsuki, Mari Fujimura, Tetsuro Nishino ....................................................... 69
A Visual Modeling and Transformation Tool for Multiple Representations of Probabilistic Behavior Model
Kai Zhao, Huaikou Miao, Jinyu Kai, Jiaan Zhou, Honghao Gao .................................. 75
Shogi Program that Selects Natural Moves by Considering the Flow of Preceding Moves
Tetsuhiko Kinebuchi, Takeshi Ito ........................................................................ 81
Estimating Ratings of Computer Players by the Evaluation Scores and Principal Variations in Shogi
Shogo Takeuchi, Tomoyuki Kaneko ........................................................................ 87
The Evaluation of Chu-Shogi’s Special rules by Using Computer Self-play Experiment
Nobusuke Sasaki ................................................................................. 93
Human-Like Build-Order Management in StarCraft to Win against Specific Opponent’s Strategies
Hiroto Takino, Kanihito Hoki ........................................................................ 99

REGULAR SESSION
Static Analysis Technique of Cross-Browser Compatibility Detecting
Sujuan Xu, Hongwei Zeng ........................................................................ 105
Static Analysis Technique of Cross-Browser Compatibility Detecting

Sujuan Xu\textsuperscript{1, 2}  
\textsuperscript{1}School of Computer Engineering and Science, Shanghai University  
\textsuperscript{2}Shanghai Key Laboratory of Computer Software Evaluating & Testing  
Shanghai 200444, China  
Xusujuan517@163.com

Hongwei Zeng\textsuperscript{1}  
\textsuperscript{1}School of Computer Engineering and Science, Shanghai University  
Shanghai 200444, China  
zenghongwei@shu.edu.cn

Abstract—Web application developers have been bothered about cross-browser compatibility problems caused by different kinds of web browsers. Existing techniques and tools commonly compare and analyze screenshots of websites rendering in different browsers. In this paper, we propose a technique for statically analyzing cross-browser compatibility problems. Our approach can easily detect whether web applications contain HTML5 incompatible features and generate a report containing the HTML5 incompatible features in the website for developers.

Keywords—cross-browser compatibility; web application; HTML5; incompatible features

I. INTRODUCTION

Web applications, used for work and entertainment, are popular in our daily lives. With the popularity of web applications, they also become more complex and richer. A web application or web app is any software that runs in a web browser. The past 20 years have seen an exceptional growth in web browsers. There are more than 100 different web browsers today [1]. Currently, people have the choice of using their favorite web browsers to run applications. However, some web applications behave inconsistently in different browsers. It is common to see that some pictures are shown in wrong position and some link buttons do not respond when clicked.

Nowadays, there are many tools and techniques to solve the cross-browser compatibility problems. Most of them focus on comparing and analyzing screenshots of applications under different browsers. These techniques require rendering all web applications in multiple browsers and a lot of manual work, so they are very costly and error-prone.

The cross-browser compatibility problems are mainly resulted from different rendering engines for various browsers which are used to implement web client technology (such as HTML, CSS and JavaScript) and render web application. Because rendering engines interpret the syntax of web’s programming languages in their own ways, the screen display may be different when the same website page is rendered in different browsers.

Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript are the primary web’s programming languages. Some features of these languages are still not supported by browsers. It is important for web application developers to know whether their applications contain these unsupported features. To research the incompatible features, we classify them according to the web’s programming language. HTML is the web’s core language. We only focus on the incompatible features of HTML in this paper.

Rendering web application optimally in accordance with the latest specification is a common goal of all web browsers. HTML5 is the latest version of HTML, which is completed on October 28, 2014 by the World Wide Web Consortium. Its goal is to replace HTML 4.01 and XHTML 1.0 established in 1999. It will make the network standards meet the needs of contemporary networks in the rapid development of Internet applications times.

Specifically, HTML5 adds a lot of new semantic elements, including \texttt{<video>}, \texttt{<audio>} and \texttt{<canvas>} which are easier to add and process multimedia and pictures in web applications. In order to rich document, it also adds some elements like \texttt{<section>}, \texttt{<article>} and \texttt{<header>}. Instead, some attributes and elements have been removed out, such as \texttt{<center>}, \texttt{<font>} and \texttt{<tt>}. At the same time, some elements are modified, like \texttt{<small>} and \texttt{hr}; some elements are adjusted, like \texttt{<a>} and \texttt{<cite>}; some elements are standardized, like \texttt{<embed>} and \texttt{<wbr>} [2].

This paper presents a new technique for detection of cross-browser compatibility. First, a HTML5 incompatible features database is built. Second, we compare the target website code with the HTML5 incompatible features database and give a report including the HTML5 incompatible features of the target web application. This method helps developers detect whether their web applications contain HTML5 incompatible features and improves web applications cross-browser compatibility.

The rest of the paper is organized as follows. Section 2 introduces related work on cross-browser compatibility detecting. Section 3 describes the system architecture of our static analysis technique and the detail method. In section 4, we use an example to illustrate our technique. In section 5 we conclude the paper.
II. RELATED WORK

A. Browser Market Share Ranking

Nowadays, all browsers are striving to implement the feature-rich HTML5. There are more than 100 different web browsers in the market today.

![Global popular browser market share ranking list in Feb. 2015](image)

Figure 1 describes the global popular browser market share in February 2015, which comes from Net Market Share [3]. In February, Internet Explorer still occupied the top ranking of the global browser which reached the market share of 57.38%. Chrome browser ranked second in market share of 24.69%. Firefox browser ranked third in market share of 11.6%.

B. Detecting Tools and Academic Research

Cross-browser compatibility is widely recognized as an important issue among web developers but hardly ever addressed directly during the software development process [4]. Currently, some cross-browser testing tools have been developed in the market, such as IETester [5], Browsera [6] and Browsershots [7].

IETester is a free tool for web developers who try to build their web applications compatible with all Internet Explorer versions. It renders a web application in Internet Explorer versions from the same interface. IETester, however, only renders a web application in different Internet Explorer versions and it has to find discrepancies in IETester’s output on human eyes.

Browsera is a commercial tool that automatically detects cross-browser compatibility. It mainly tests and reports cross-browser layout difference and scripting errors on website. Instead of having to look for discrepancies in web applications on human own, Browsera is like a tester, finding visual inconsistencies between different browsers and giving a report which contains potential problems.

Browsershots is a free open-source online web application providing developers a convenient way to test their website’s browser compatibility in different operating system. Developers can choose Linux, Windows, Mac as operating system and select Firefox, Opera, Chrome, Internet Explorer and several others browsers with their versions.

Academic study of cross-browser compatibility detecting is just beginning in recent years. Choudhary and Prasad [8] propose a comprehensive approach for cross-browser incompatibilities (XBIs) detection and classify XBIs into three main types: structure, content, and behavior. They prove that three incompatible types are independent, and each of XBIs is detected by using different testing technique. Their research helps developers accurately identify cross-browser incompatibilities in web applications.

Li and Zeng [9] focus on detecting whether the behaviors of the target web pages are lost in various browsers. The paper presents a model of interaction transition to describe the behavior of web application.

The existing cross-browser compatibility tools, both commercial and academic, require web applications rendered in browsers and analyze the screenshots of applications under different browsers. Our proposed technique directly detects web application code, as we discuss in the next section.

III. SYSTEM ARCHITECTURE

Static detection technique is an important method of web application testing. Analyzing the code directly and not actually running the program being tested are the basic characteristics of static test [10].

We study cross-browser compatibility problem from a new perspective. Now, there are several tools which can detect how well various browsers support the HTML5 standard. We use those tools to collect the HTML5 incompatible features. Then we detect whether the target web applications contain the incompatible features. Figure 2 presents the system architecture of our static analysis approach.

![System architecture of cross-browser compatibility detecting](image)

The approach consists of two phases. The first phase builds a HTML5 incompatible features database. We choose three most popular browsers and extract their current unsupported HTML5 features into tables. The cylinder of figure 2 is the incompatible features database which consists of elements and attributes that are not supported by our tested browsers.

The second phase detects whether the target web application contains HTML5 incompatible features and outputs a cross-browser incompatibilities report to the web application developers. Our method needs web developers provide the source code of the tested web applications, as is shown in the first folded angle rectangle of figure 2. After detecting web applications by our technique, the developers will get a report, as is displayed in the second folded angle rectangle of figure 2. The following sections present further details of these phases.

A. Creating Incompatible Features Database

Our approach extracts HTML5 incompatible features of tested browsers by an online tool called HTML5 TEST [11] which is to detect how well the browser used support the HTML5 standard and related specifications. To test a web browser, we only need to visit the home page of the website html5test.com. The application returns a score out of 555 points and shows comparison tables of HTML5 features in detail. The score is calculated by testing for the many new features of HTML5. Each feature is worth one or more points.
We classify unsupported features into two types: unsupported element and unsupported attribute. An unsupported element denotes that all the attributes and methods of this element are not supported by the tested browser. While an unsupported attribute denotes that the attribute or method of the tested element is still not supported by the tested browser. For example, the <main> element is unsupported by IE 11, so we define it as a unsupported element; The download attribute on the <a> element is still not supported by IE 11 and there are some attributes on the <a> element are supported by IE 11, we define the download attribute of the <a> element as an unsupported attribute.

Through accessing the website html5test.com, we collect the unimplemented HTML5 elements of the tested browsers and create a HTML5 incompatible features database. Because the score of IE is minimal, we choose IE as example to explain our approach. The detailed process of building database is presented as follows.

The first step collects the unsupported features of IE. When IE visits the home page of the website html5test.com, we can find out which parts of the HTML5 are supported by IE. Figure 3 shows some supported and unsupported HTML5 features by IE. For example, the <main> element following the word “No” denotes that the element is not supported by IE. While the <figure> element following the word “Yes” denotes that the element is supported by IE. Because supported features are far more than features that are not supported, we only collect unsupported features from the page.

We classify unsupported features into two types: unsupported element and unsupported attribute. An unsupported element denotes that all the attributes and methods of this element are not supported by the tested browser. While an unsupported attribute denotes that the attribute or method of the tested element is still not supported by the tested browser. For example, the <main> element is unsupported by IE 11, so we define it as a unsupported element; The download attribute on the <a> element is still not supported by IE 11 and there are some attributes on the <a> element are supported by IE 11, we define the download attribute of the <a> element as an unsupported attribute.

The second step creates two tables used to store unsupported features. We add all unsupported elements to a table named IE_unsupported_elements and all unsupported attributes to the other table named IE_unsupported_attributes, as is shown in figure 4.

According to the above method, we extract unsupported features by Chrome 39 and Firefox 35 into tables respectively. In the end, we create a HTML5 incompatible features database, which consists of six HTML5 incompatible features tables.

B. Detecting Incompatible Features

Most of web applications are created in the standard markup language HTML nowadays. Web browsers can read HTML files and compose them into visible or audible web pages. HTML consists of tags enclosed in angle brackets, such as <html> and <body>. Although there are some tags are unpaired like <br> and <img>, most of HTML tags come in
pairs, for example <h1> and </h1>. The first tag in a pair is the start tag and the second tag is the end tag.

Regular expression is mainly lexical analysis by pattern matching of characters, which describes a set of strings that can be used to check whether a string contains a certain substring or replace a matching substring.

**Algorithm 1: Detect Incompatible features**

**Input**: C: Target web application code  
T1: unsupported elements table  
T2: unsupported attributes table  
**Output**: X: List of XBIs  
**Begin**  
X ← ∅  
While scanningCode (C) ≠ NULL do  
e ← findElement(C)  
if (matchElement(e, T1)) then  
add(r, e, X)  // r is the row of e in C  
else if (matchAttribute(e, T2)) then  
add(r, e, a, X)  // a is the attribute of e  
end  
end  
return X

Algorithm 1 presents our technique of detecting incompatible features. The algorithm’s input consists of the code of the target web application and the two tables containing unsupported HTML5 features by the tested browser. The algorithm’s output is a list of XBIs of the target web application.

Our technique mainly uses regular expression to detect HTML elements of the target web applications. It defines some complex regular expressions which are used to identify different HTML features. Our method only analyzes the start tags enclosed in angle brackets of the target website. When it finds out an incompatible element, it will output the row number and the start tags.

Our algorithm consists of two steps. The first step identifies the element e of the target web application. This is implemented by function findElement() in Algorithm 1. The second step compares the element e found in the first step with the unsupported features tables T1 and T2 to check whether the tables contain e. This operation is implemented by function matchElement(e, T1) and matchAttribute(e, T2). The method will iterate the above two steps until the last character of the target web application is detected.

**IV. CASE STUDY**

In this section, we introduce a motivating example to illustrate our approach. The example is a website for papers introduction, and the HTML page is shown in Figure 5.

In our example page, the <body> tag contains the main web page elements to be rendered. The input date box is used to choose date (line 8). The text enclosed in an <summary> tag is the title of the paper (lines 11-13). The text enclosed in an <p> tag is abstract of the paper (lines 14-18). The “href” attribute of the <a> element denotes the position of the paper, and the “download” attribute allows user to download paper directly from browser (lines 21-24).

We use the above section algorithm 1 to detect whether the example page contains HTML5 features that are not supported by our tested browsers. Figure 6 is the lists of XBIs of our technique detection. There are 4 incompatible features by IE 11 in figure 6(a). The first incompatible feature is the “date” attribute of the <input> element, which is in the eighth row of the page. The second incompatible feature is the <details> element. The third incompatible feature is the <summary> element. Because the <details> element is used with the <summary> element, the two elements can be seen as an XBI. The fourth incompatible feature is the “download” of the <a> element. There are 3 incompatible features by Firefox 35 in figure 6(b). Because all features of the example are supported by Chrome 39, we do not show its result in figure 6.

To test the effectiveness of our technique, the example is rendered in the tested browsers. Figure 7 is the screenshots of the example page three browsers.

Comparing above three screenshots in figure 7, we can observe three XBIs.

1. The first XBI is the date input box. When firstly rendered in three browsers, the box has the same display. However, when click the box in three browsers, only Chrome browser correctly display its function. IE and Firefox do not support the “date” attribute of the <input> element.
(2) The second XBI is display of title and abstract of the paper. This function is implemented by the interactive elements <details> and <summary>. When the page firstly rendered in the browser, it should only display the text enclosed in the <summary> element, as is show in figure 7(c). Only user clicks the title of the paper, its abstract could be shown to user. IE and Firefox both do not implement this function.

(3) The third XBI is the function of download paper. When we click the link “Download Paper” in three tested browsers, Firefox and Chrome both could download the paper to user. IE still does not support the function.

By contrast, the incompatible features detected by our technique are accordant with the comparison of the screenshots of the tested browsers.

V. CONCLUSION AND FUTURE WORK

Web application developers have been troubled by cross-browser compatibility issues for a long time. This paper presents an approach for static analysis technique of cross-browser compatibility issues that solves the problem from a new perspective. To do so, we build a HTML5 incompatible features database for three tested browsers and propose an algorithm to detect incompatible features in the target web applications. Finally we use an example to illustrate the effectiveness of our approach. Currently, our database only contains the incompatible features which detected by HTML5 TEST. In the future, we will enlarge our database and improve the efficiency of our algorithm.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (NSFC) under grant No. 61073050 and No. 61170044.

REFERENCES

[7]. Browsershots, http://browsershots.org