MARCO: MARyland browser COnfiguration based testing tool

Vijay Gopalakrishnan & Srinivasan Parthasarathy

21st December 2001

{gvijay,sri@cs.umd.edu}

Abstract

In this paper, we address the problem of testing websites for correctness under various browser configurations. Most existing web testing tools and approaches do not go beyond checking links, spelling and HTML tags. Our approach is tailored for revealing webpage errors which are manifested only under certain browser configurations. We capture the interaction between the components of a webpage and a browser configuration, in an information base. This information base is used by our testing tool, MARCO, to identify potential problems with a webpage. We demonstrate the viability of our approach by analyzing three real world web sites. We believe our tool would be of use to web developers, including browser developers and website developers.

1 Introduction

The distributed nature of the World Wide Web is undoubtedly the most important reason for its prominance. The Web provides a convenient platform for various technologies and applications to co-exist. Web applications are characterized by massive user population, varying needs of users and the diversity of environments. It is precisely these characteristics which present new challenges in the area of testing web applications.

Of particular interest to us is the dependencies between the components of a web page and the browser configuration. The correct behaviour of a webpage is dependent upon the correct behaviour of all the components contained in the webpage. The correct behaviour of these components, in turn, is affected by the actual browser configurations at the client side. Typical web pages today include components such as HTML, JavaScript, Java, cookies, multimedia content, security information, etc. Most of these, if not all, can be directly affected by the various settings in a browser. For instance, a security protocol which is entirely dependent on the JavaScript code contained in the webpage might fail because the user had turned JavaScript off.

We motivate our problem further with an example. Let us assume that we need to test a webpage: www.cs.umd.edu. We are interested in its behaviour under Netscape in Linux. This is a webpage which uses JavaScripts for downloading images. However, if JavaScript is not enabled at the client side then it downloads a particular image by default. Hence, it does not depend upon JavaScript heavily as far as its functionality is concerned. Yet disabling JavaScript adversely affects fonts and colors across the entire webpage. In other words, errors which cannot be directly attributed to a particular component in the webpage (in this instance, JavaScript is not responsible for fonts or colors) might be revealed when tested under various browser configurations (maybe, due to wrong browser inplementations).

Our approach to this problem involves two steps. We first study the interaction between various components in a webpage with various browser settings. For example, in the example above, we observed the behaviour of the page under JavaScript enable/disable settings with Netscape and Internet Explorer (IE) under Linux, Windows and Solaris. In the second step, we create a tool, MARCO, which can automatically analyze webpages and report potential problems under various browser configurations. This tool makes use of the information gathered in step one.

The outline of the paper is as follows. In section two, we discuss existing web testing approaches. Section three discusses the design of MARCO. Section four discusses the implementation details of MARCO and evaluation of MARCO. We conclude with some remarks and future work in the last two sections.

2 Existing Web Testing Approaches

There are a wide range of web testing tools which are currently available. The following are the various categories of web testing tools.

- Load and Performance Test Tools
- Java Test Tools
- Link Checkers
- HTML Validators
- PERL and C Programs for Validating and Checking
- Web Functional/Regression Test Tools
- Web Site Security Test Tools
- External Site Monitoring Services
- Web Site Management Tools
- Log Analysis Tools

- Other Web Test Tools

However, none of these tools and approaches really solve the issue which we address in this paper.

Perhaps, among the existing tools, the one which comes closest to our needs is the NetMechanic tool. This tool analyzes a given webpage and reports the problems with various browsers. However, even this does not check for various browser configurations or operating system dependencies.

3 Web Testing using MARCO

In this section, we discuss our approach in detail. Our approach consists of two parts.

- Collecting information about dependencies between web components and browser settings (Infobase).
- Using this Infobase to test web pages for potential problems (MARCO).

In this way we can avoid the overhead of manually testing every web page against every possible browser configuration.

3.1 InfoBase

The InfoBase captures the interaction between various web components and browser settings. For ease of visualization, we may think of the InfoBase as a matrix whose rows correspond to the various web components and whose coloumns correspond to the various browser configurations. The components include JavaScript, Cascading Style Sheets, Images, Cookies, etc. The browser configuration includes the the values of all the user defined options in the browser, the type of browser (Netscape, Internet Explorer, etc.), the browser version and the Operating System on which the browser is user.

Each entry in the InfoBase indicates if a particular component is compatible with a particular browser configuration. Thus, revisiting our JavaScript example, we might say, the component JavaScript is not compatible with all the browser configurations for Netscape where JavaScript is disabled.

An important point needs mention here. Although the goal of our tool is to automate configuration based web testing, the intermediate step of building the InfoBase is not automated. The main reason which prompted our decision to manually build the InfoBase was that there are no automatic and standard ways of setting browser configurations and checking for correctness of the displayed web page.

3.2 MARCO

MARCO is an interpreter for webpages. It takes as input a web page, analyzes the various components in it and reports on potential problems. MARCO parses the input webpage and creates a stack of the various tags and attributes in the web page. The idea behind creating a stack is that we can detect malformed tags immediately. Thus, if there is a closing tag without the corresponding opening tag, we can detect this without any additional overhead. For every tag and attribute, MARCO consults the InfoBase to see if there is a potential problem of interoperability with certain browser settings. All such entries which are relevant to the web page are recorded in a log file.

4 Implementation and Evaluation

MARCO is implemented as a c++ library. The main components of the library are

- A custom built interpreter for web components
- Attribute and Tag analyzer
- InfoBase

The main function of the interpreter is to parse the given webpage and log syntactic errors in the page. The interpreter constructs a stack of the various tags in the page. Thus an opening tag without a closing tag or viceversa can easily be detected. Also our interpreter can detect errors of the form where the order of the closing tags is different from the opening tags. Note that although this error might be acceptable to many browsers, it can lead to other errors in the future when the developer modifies the contents of the webpage.

The tag analyzer, as the name indicates, analyzes the various tags and logs potential browser compatibility issues. There are two parts to the tag analyzer. The first part, simply detects if the given tag is incompatible with a particular browser. There are several tags in a webpage which are specific to browsers. Netscape defines a few HTML tags which cannot be interpreted by Internet Explorer. Microsoft defines a lot of HTML tags which cannot be interpreted by Netscape. Sun has its own share of proprietary tags. All such tags are detected and logged as a potential error in the log file.

The second part is a little more involved. For every tag in the page, the analyzer detects if there is a potential problem with a particular setting of a given browser. For eg., if the tag under consideration is a JavaScript tag then there is a potential problem with all settings of the Netscape browser where JavaScript is disabled. There are many examples of this nature. All such occurences are logged in the log file.

Both the parts of the tag analyzer make use of the information available in the InfoBase. Infact, the main purpose of the InfoBase is to provide a lookup mechanism for the analyzer. We will look at the details of the InfoBase in the next subsection.

4.1 InfoBase

The InfoBase stores the dependencies between components of a webpage and configurations of a browser. As mentioned earlier, it can be visualized as a matrix whose rows correspond to the webpage components and coloums correspond to the browser settings and the entries indicate if a particular webpage component is compatible with a particular browser setting.

The InfoBase was generated manually by testing individual webpages against various browser configurations. The various browsers include Netscape (versions 4.7 and 6.2) in Linux, Solaris and Windows and Internet Explorer (version 5) in Windows. The various web components include JavaScript, Cascading Style Sheets (CSS), Images, etc. Below we present a portion of our findings which are part of the InfoBase. We do include all our findings due to space constraints.

- When JavaScript is disabled in Netscape 4.7 in Linux and Solaris, fonts which are outside the scope of the JavaScript and hence should not be affected, indeed get affected. This negatively impacts the *look and feel* of the webpage.
- Disabling CSS in Netscape 4.7 in Linux and Solaris affects the rendering of the fonts in a page. The default font options in these browsers are not exercised.
- If the user has disabled images, the alternate text is made visible in Internet Explorer. Netscape does not show the alternate text in any of the operating systems.

These are some of the entries in the InfoBase which are made use of by our tool to report on potential errors.

4.2 Sample log

Below we present a typical output generated by our tool in a test run. We tested our tool with three real world web pages, www.cs.umd.edu, www.umd.edu and www.cs.umd.edu/~atif/Teaching/CMSC838M.html. The following is the summary of results, when the tool was run against the third page.

- MARCO has detected the use of one(or more) of the following kinds of tags 1. LOOSE tags 2. Proprietary tags 3. Unknown tags Such tags can cause difference in the display on differnt browsers.
- MARCO has detected the presence of unmatched tags. Some browsers may be lenient on these, while some may not be. You may be getting the error because you do not open and close the tags in the correct order.
- Marco has detected the use of FONTS tag with SIZE as an attibute. The size specified by you is an abstract size. Each browser renders the same abtract size differently. So the page looks different on each browser. We advice you to use Style sheets instead.

- Marco has detected the use of IMG tag without ALT attibute. This could create accesibility problems if the user has disabled getting images. Some browsers, however, do not do a good job of diplaying ALT text! We also do not recommend having links using images due to the above reason.

While we have presented only a summary here, the browser logs the exact lines where the errors have occured. Thus we see that MARCO can be used to alert web developers in advance about the potential problems which could be caused by their web pages on various browser settings.

5 Future Work

Several extensions are possible for MARCO. The first is extending the scope of the InfoBase. Its possible to add more information to the InfoBase by considering more browsers and browser versions. It is also possible to add more information by considering more web components. Currently MARCO analyzes only HTML tags. Although it logs JavaScript tags, it does not does not analyze within the JavaScript itself. Also, XML tags and applets are ignored. These are certainly worth considering.

Another important extension to MARCO is integrating our InfoBase with browsers so that browsers can make intelligent decisions when downloading webpages. For instance, if the page requires JavaScript to be turned on and if the current configuration has JavaScript disabled, the browser may be able to prompt the user for some change in the settings. We believe this might result in substantially improved quality of rendering of webpages in many browsers.

There is another interesting way of improving MARCO's usability. Currently MARCO takes individual webpages as input. We could allow MARCO to take the address of a website as input. In this way, it could analyze all the pages associated with the webpage, including the pages corresponding to the links in the webpage. Although, this is not conceptually a significant extension to our tool, we believe this could lead to a substantially improved usability.

6 Conclusion

In this paper, we have proposed, designed, implemented and evaluated a novel approach for testing web content. Our approach involves the use of a manually generated information base which captures the interaction between various webpages and browser configurations. We then use this information base to automatically analyze webpages to report potential problems with browsers in their various settings. To our knowledge, ours is the first web testing approach which takes into account various browser configurations.

We believe our tool will be of immense use to web developers, including website developers and browser designers. Our tool could be used by website developers to assess the quality of their site. Our tool can also be incorporated as part of existing browsers, to warn the user about required changes in configuration for the correct rendering of the webpage. We have also proposed several extensions which could lead to substantial developments to the usability of our tool.

7 References

1. www.netmechanic.com
2. www.w3c.org
3. www.internetqa.com
4. www.evalid.com
5. www.soft.com