IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP5102

Identifying and Addressing the Extra Issues Involved in Assuring Quality of Client-Side Reflective and Dynamic Web Applications

M. Sh. Aun
Nagoya University, Dept. of Information Engineering, Nagoya City, Japan, Sharaf@agusa.nuie.nagoya-u.ac.jp

S. Yuen
Nagoya University, Dept. of Information Engineering, Nagoya City, Japan, yuen@nuie.nagoya-u.ac.jp

K. Agusa
Nagoya University, Dept. of Information Engineering, Nagoya City, Japan, agusa@nuie.nagoya-u.ac.jp

ABSTRACT

This paper identifies the extra issues involved in assuring the quality attributes of Client-side Web Applications (CSWAPs) and then presents an approach for addressing such issues. The approach combines static (preprocessing) and dynamic processing together with features engineering techniques. The effectiveness of our approach is in the sense that, it provides opportunities for better analysis that can facilitate the debugging of client dynamic-enabling components and the validation process or conformance to standards of HTML components. The inherit advantages of our approach enables it to be helpful for assuring several other quality attributes.

INTRODUCTION

Web applications, nowadays, impose some entirely new challenges in the world of software quality. Assuring their quality attributes involves several extra issues over assuring the quality attributes of stand-alone traditional applications. These extra issues are due the fact that Web applications differ from traditional software applications in several critical dimensions [3]. Moreover issues like the coexistence of multiple technologies in an application, the immediate interpretation feature, the reflectiveness property of the languages used, the high volatility and the unpredictability of the Run-Time Environments (RTEs) used have lead to the fact that, trying to make a pure transposition of quality assurance techniques (like other techniques [10]) from software engineering to the Web engineering is both difficult and inadequate.

To support the design phase, several frameworks, architectures, and models, such as those described in [2] have already been designed and proposed. As for other phases, only few works like [4], [5], [6], and [7] focus on some aspects related to automated data collection and data analysis considering the client-side as pure presentation layer. In fact researches [8] have shown that no automated support exists yet for anything more than basic HTML editing and barely any CASE tool supports advance Web-based applications development. Thus, several issues still have to be studied well and more specific approaches are required for assuring quality attributes of Web applications.

This work represents an on-going effort for developing a computer-aided environment to support the construction of high quality Web applications. In this paper, we identify the specific extra issues involved in assuring the quality attributes of the CSWAPs over stand-alone applications and then present an approach for addressing such issues. The aim of this work is at identifying the extra issues involved in assuring the quality attributes of CSWAPs and on the set of components that incorporate the essential architecture design of an environment dedicated for addressing such issues. The focus is on the client-side applications assuming the result of any computation at the server-side to be web pages that have to be sent and presented to the client.

In this work we go beyond our previous work [9] by scaling it up to deal with a wide range of applications and provide opportunities for better analysis that can not only support debugging but also conformance to standards. In addition to combining static (preprocessing) and dynamic analysis, we exploit the benefits of features engineering [11] in order to deal with a wide range of applications.

Section 2 identifies the extra issues involved in assuring the quality attributes of CSWAPs. In Section 3, our approach for addressing such issues is described. Section 4 puts the proposed approach in practice. Section 5 discusses the characteristics of our approach and Section 6 compares it with other related works. Finally the paper is concluded with a conclusion and future directions.

IDENTIFYING THE EXTRA ISSUES

Assuring quality of CSWAPs involves several extra issues over assuring traditional stand-alone applications. These issues may include the following:

- The methodology has to be able to fill the analysis gap due to the lack of compilation and the type-less property of the languages used. Client enabling components are usually immediate interpreted. Immediate interpretation means, traditional compiling techniques are not available resulting in lack of type checking techniques, for example. This also means that bugs or other failures are only noticed during run-time. Moreover, Web scripting languages, like other scripting languages [12], are usually type-less. This implies that all variables look and behave the same so that they are interchangeable. Such feature has penalty: at time we think a variable or an expression has a certain type or data in it, when in truth, something entirely different in there.
- The methodology has to be capable of overcoming the challenges introduced by the reflective and dynamic properties of the applications. Due to the first property, a program can change itself or generate new ones making the state of the program to be a combination of both the global variables and the program itself. Web RTEs also are much more volatile as web changes tremendously over the course of a few milliseconds. More

concurrent activities are involved on the web. The more dynamic the application is, the more challenges in assuring its quality.

- The methodology has to provide some mechanisms for controlling0program execution so that the developer can investigate the state of0the program at certain points during execution. Such mechanisms have to be platform-independent.
- The methodology has to be capable of separating useful features out of the implementation artifacts dealing with the complexity introduced by the coexistence of multiple technologies on the web. A primary benefit to be gained from separating features out of the implementation artifacts is to facilitate both debugging and validating client-enabling components of the application under consideration.
- Another specific and very important requirement is that, the methodology should be able to cooperate with its surrounding environments such as the browsers and be able to coordinate it activities with them. Scripting objects, as an example, are mainly environmental objects rather than language created objects. Thus, any failure due to manipulating them can't be treated far from the environment, which creates such objects.

In addition to these issues and in order to deal with wide range of applications, the methodology has to be capable of restructuring the base-code. Unlike traditional programs, the source code of CSWAPs can be scattered over and generated at several layers of the Web architecture.

THE APPROACH FOR ADDRESSING THE ISSUES

Having introduced the extra issues in assuring the quality of CSWAPs, we now proceed to describe our approach for addressing such issues. To address the challenges and to fulfill the specific requirements, our approach combines four main complementary sets of techniques. These are: features separation, preprocessing, dynamic processing, validating, and base-code restructuring techniques. This paper focuses on the first three techniques introducing others for the purpose of completeness. We assume the result of any computation at the server-side to be web pages that have to be sent and presented to the client.

3.1 The Approach Reference Model

The approach reference model consists of five main complementary and interacting components, Fig. (1), and will operate with the guidance from the developer. These components are: features separator component, preprocessing component, dynamic processing component, validation component, and base-code restructuring component. Dynamic preprocessing component involve: debugging and testing library functions, interpretable and code extended code re-generator, and interpreting facilitator sub-components. Each of these components has its own purpose and function as described briefly next. More detailed description will be presented in Section 4.

3.2 Components and Their Functionalities

3.2.1 Features Separator

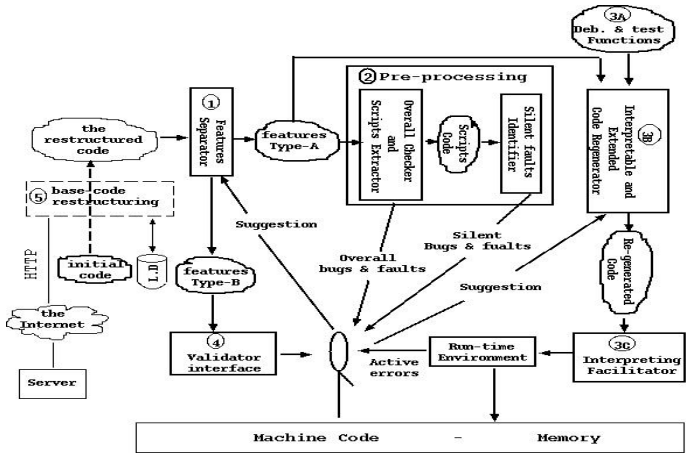
This subsystem is responsible for identifying and separating useful features out of the implementation artifacts of an application. A primary benefit to be gained from separating features out of the implementation artifacts is to facilitate the debugging process of client dynamic-enabling components.

Two categories of features represent the output of this subsystem. The first category (type-A) is those features that contain dynamic-enabling components (i.e. scripts together with their related rendering components). The second category is those features with no scripts. This categorization is necessary, as those features containing dynamic components must be treated differently while assuring quality.

3.2.2 Preprocessing

Preprocessing is necessary for filling the analysis gap due to the lack of compilation. As indicated above, client-enabling components are usually immediate interpreted. In addition to help in eliminating both overall and silent bugs[9], preprocessing can serve other purposes.

Figure 1. Approach reference model



These may include code beautification, documentation and checking for cross-browsers scripting.

This subsystem involves two subcomponents. These are:

- ***Overall Checker and Components Extractor***: Features that contain dynamic-enabling components represent the input to this subsystem. The source code of the whole feature should be read, overall checked, and the embodied component (such as script code) should be extracted. The output is the script components in addition to overall bugs identified in a feature. Compared to available checking tools such as weblint [13], this subsystem has also to extract the embodied dynamic-enabling component for further processing.
- ***Silent Faults Eliminator***: This component concentrates on eliminating silent bugs [9]. Such category causes the RTE neither to run the application nor to produce any helpful error messages. In such case pre-processing is needed as such category of bugs cannot be eliminated by other ways.

3.2.3 Dynamic Processing

Dynamic processing serves several purposes. It identifies and eliminates active or behavioral bugs, provides the necessary mechanisms for controlling program execution while debugging, and helps in overcoming the challenges due to the dynamic property of the applications. It involves three subcomponents. These are:

- ***Library Functionalities***: Devoted for debugging functionalities realized as functions that can be augmented within the source code during the expansion process. The functions must include specialized code for inspecting both objects and scalar variables.
- ***Interpretable and Extended Code Re-generator***: The inputs to this part are the source-code of the features containing scripts, the required debugging functions and the developer options (suggestions). The source code is to be expanded and the debugging functions to be augmented. Developer options are necessary for providing more flexible debugging options. The output of this part is the extended code, which is the original source code plus the required functionalities that facilitate the debugging.
- ***Code Interpreting Facilitator***: This part is necessary for facilitating the execution of the applications directly from the context of the debugger. It works as an interface between the extended code and the interpreter.

3.2.3 Validating Interface

This component passes the source code of the input features (type-B) through a special program that compares the code against standards-based measures. It checks syntactic accuracy, structural integrity and conformance to standard requirements. With separating of features, many exiting and emerging tools by W3C and others can be utilized for

assuring conformance to standards. Nowadays, for example, there are several available tools like [14] by W3C to check HTML documents, others like [15] to make them tidy, etc.

3.2.4 Base-code Restructuring

Unlike traditional programs, the source code of CSWAPs can be scattered over and generated at several layers of the Web architecture. Therefore, to deal with a wide range of applications, the approach must involve the use of code restructuring techniques. This is the main purpose of this subsystem.

4. THE APPROACH IN PRACTICE

To prove the effectiveness of our approach, we intended to provide some examples clarifying the concepts with the help of a prototype-debugging tool being implemented to work within the context of the approach. However due to the lack of space, we provide more detail explanations regarding the techniques involved.

4.1 Features Separation

Due to the different ways by which the code of Web applications is organized, several main categories of features can be identified and separated. Sometimes features are encapsulated in a page or a set of pages. Other times, a single page may include code associated with multiple features. Yet another cases, code contributing to a feature may be found at the client, at the server, or it may be split between the client and the server. For this reason, several categories of features can be identified. These are: *features tangled in client-side, features tangled in server-side, and features tangled across client and server sides.*

Fortunately, from th e viewpoint of CSWAPs, an application can be regarded as a set of static/dynamic Web pages that are usually written in or dynamically created in HTML code containing scripts and other client enabling technologies. In contrast to procedural applications, where a specific functionality is often identified with a subsystem or module, the functionality in client-computation of Web applications comes from the interacting of pages. Accordingly, there are two main categories of features. These are:

- features tangled in a page.
- features tangled across pages.

4.1.1 Identifying Features Tangled in a Page

Web pages usually consist of code that contain a variety of components (such as scripts, Applets, HTML components, images, audio, etc.), which are usually realized with different categories of languages. Therefore, identifying features tangled in a page can be accomplished by treating a page as a collection of fragments rather than a single entity. These fragments may include:

- fragment type-0: only mark-up components.
- fragment type-1: a script component plus its related rendering components.
- fragment type-2: an Applet component plus its related rendering components.
- fragment type-3: mark-up components plus a mix of related components such as an Applet plus a script that communicate with each other.
- fragment type-4: other components such as video, audio, etc.

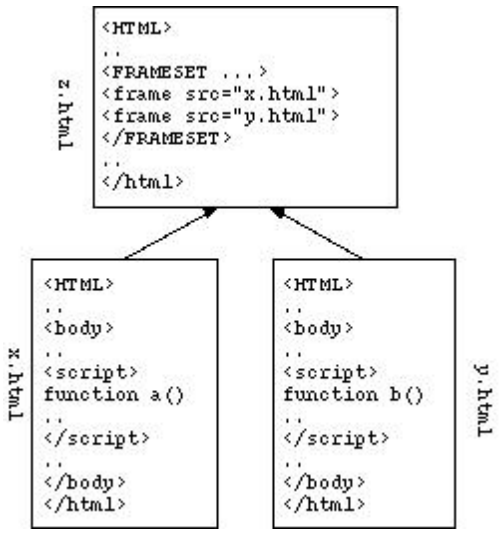
Therefore, the existence of multiple technologies in a web page can be considered as the key issue for separating of features tangled in a page.

4.1.2 Identifying Features Tangled Across Pages

For the case where features can be tangled across pages, Fig. (2) shows an example.

In this example, there are two features tangled across pages. These features are: (1) combination of page x plus a subset of page z where x appears and (2) combination of page y plus a subset of page z where y appears.

Figure 2. Features tangled across pages



4.2 Preprocessing

An example where the RTE neither tells what is wrong nor executes the script (i.e. a silent bug) is when a reserved word is used as a function name, like the word *case*.

4.3 Dynamic Processing

Dynamicity at the client, which is mainly caused by script components, is adequately managed as the subsystems can cooperate their activities with the RTE. The approach can augment the required functionalities for assuring the quality attributes to the components themselves. This allows them to be uploaded by the interpreting facilitator without the need for extra mechanisms or techniques.

4.4 Base-code Restructuring

Currently, this subsystem is still under development but its subcomponents have been identified. These are code re-organizer, page classifier, and code re-factoring components. Details related to this subsystem will be a topic of subsequence paper.

5. DISCUSSIONS

5.1 Addressing the Issues

The new proposed approach consists of five main (or seven sub.) components (Fig.1) each of which has its own purpose and function. By looking closely at these components, we will find that they, collectively, fulfill table (1) the specific requirements and address the issues presented in Section 2.

The first component (or features separator) is responsible for dealing with issue-4. It enables a feature to be separated overcoming the difficulties introduced by the fact that on the web multiple technologies have to coexist in one application. The second component (or preprocessing) mainly responsible for making overall checking of the features containing scripts and eliminate silent bugs [9]. Together with the validation component, it can address issue-1, presented in Section 2. This overcomes the challenges due to the immediate interpretations and the lack of the debugging functionalities of the RTE. The third component (dynamic processing), which consists of three sub-components, serves several purposes. It provides the necessary mechanism for

Table-1. Addressing the issues.

	Comp. 1	Comp. 2	Comp. 3A	Comp. 3B	Comp. 3C	Comp. 4	Comp. 5
Issue – 1		O				O	
Issue – 2			O	O	O		
Issue – 3		O	O				
Issue – 4	O						
Issue – 5			O	O	O		
Additional							O

controlling program execution, facilitates the execution of the applications from within the context of the new environment, and also implies the solution for the difficulties, which arise due to the reflectiveness and dynamic properties. The fourth component is for validating features that contain no scripts from different point of views such as conformance to W3C standards recommendations. Finally, the fifth component is to enable the approach to deal with a wide range of applications.

5.2 The Strength of the Approach

The strength of the proposed environment is in the sense that, it is target-able for assuring several quality attributes of the CSWAPs rather than only debugging and validating. Although, the primary benefit to be gained from separating features out of the implementation artifacts is to facilitate both debugging and validating, the inherit advantages of the approach enables it to be helpful for assuring several other quality attributes.

Moreover, the new approach is not tied to any specific language, platform or RTE. This is the case as the set of components, incorporating the essential architecture design, can be used as a guideline for the development of tools to handle CSWAPs applications built with different languages and running on different platform. The methodology requires no modification or extension of the current available RTEs.

6. RELATED WORK

To our knowledge, there is no work in the academic literature on quality assuring techniques targeting the dynamic behavior of the web. Exception for that is one paper by the <*Bigwig*> project team (Claus Brabrand et al.) [16], which addresses the static validation of dynamic generated HTML in the context of the <*Bigwig*> language. While the aim of their work was at validating HTML components in the context of specific language, our work aim is at addressing the extra issues, which are involved in assuring the quality attributes of CSWAPs focusing on scripts, as they are the main reason behind making the web more dynamic.

Another paper (tailored to supporting maintenance) by P. Atzeni, et al. [17], has proposed a model that can support maintenance as well as the design phase of Web applications. Another paper (tailored to testing) by H. M. Kienle et al. [18], have introduced a classification of Web analysis techniques. Another paper (tailored to architecture recovery) by A.E. Hassan et al. [5], have proposed an approach for recovering the architecture of Web applications on the windows platform. Jonas Boustedt [6] has made some efforts trying to find suitable criteria for classifying equivalence pages or nodes for the server responses in Web services with the help of spider.

These efforts, however, in addition to be tailored to specific tasks, have considered the client as a pure presentation layer. Our work differs from such efforts in presenting a new approach for facilitating better analysis that can overcome many challenges including the complexity introduced when the client works beyond the presentation layer.

The software engineering literature includes some works on assuring quality, but it is generally aimed for non-web or tradition software systems rather than for web applications. In the web engineering literatures including the proceeding of the five international workshops [1] organized and the two IEEE multimedia special issues, there is no paper yet investigating assuring quality of CSWAPs.

7. CONCLUSIONS AND FUTURE WORK

In response to the lack of existing approaches specifically designed for identifying and addressing the specific extra issues involved in assuring the quality attributes of CSWAPs, we have presented our approach for that purpose. We focused on identifying the extra issues involved in assuring the quality attributes of CSWAPs and described the architecture design of the approach for addressing such issues.

It was found that, several extra and specific issues are involved in assuring the quality attributes of CSWAPs. These issues include the need for fulfilling the analysis gap due to the lack of compilation and the typeless properties of the languages used, the need for overcoming the challenges of the dynamic property of the applications, the need for program execution controlling mechanisms and the need for reorganizing and restructuring the base-code to enable a feature to be identified and separated.

To address such issues, our approach combines static (preprocessing) and dynamic processing together with features engineering techniques. Preprocessing is necessary for filling the analysis gap due to the immediate interpretation. Dynamic processing serves several purposes. It provides the necessary mechanisms for controlling program execution, and deal with the dynamicity and the reflectiveness properties. To deal with a wide range of applications, the approach involves features separating techniques.

This work highlights several interesting research issues. Assuring quality was primarily performed through facilitating debugging and conformance to standards. Although there are many other quality attributes to assure (such as understandability, maintainability, and so on), the inherit advantages of our approach enables the environment to be target-able for assuring such quality attributes.

ACKNOWLEDGMENT

The authors would like to thanks S. Yamamoto (the associate professor in Aichi Prefectural University), T. Hamaguchi (the research associate in the Lab.) and all members of Agusa Lab. for their useful comments.

REFERENCES

[1] Ginige, A. and Murugesan, S. (2001) “Web engineering: An introduction”, IEEE Multimedia, 8(1), pp.15-18.

[2] Costagliola, G., Ferrucci, F. and Francese, R. (2002) “Web Engineering: Models and methodologies for the design of hypermedia applications”, in “Software Engineering and Knowledge Engineering hand book”, vol.2 Emerging Technologies, World Scientific Publishing Co., pp.181-199.

[3] Isakowwitz, T., Stohr, E. and Balasubramanian, P., (1995) “RMM: A methodology for structured Hypermedia design. “, Commun. ACM 38(8), pp.34-44.

[4] Tesoriero, R. and Zelkowitz, M., (1998), “A web-based tool for data analysis and presentation”, IEEE Internet Computing, pp.63-69.

[5] Hassan, A. and Holt, R., (2002), “Architecture Recovery of Web Applications”, in Proc. of ICSE 2002: International Conference on Software Engineering, Orlando, Florida, pp.19-25.

[6] Boustedt, J., (2002), “Automated analysis of dynamic web services”, Master thesis in computer science, Uppsala University, Sweden.

[7] Ricca, F. and Tonella, P., (2001), “Building a tool for the analysis and testing of Web applications: Problems and solutions”, Proc. of TACAS’2001, Genova, Italy, LNCS 2031 pp.373-388.

[8] Barry, C. and Lang, M., (2001), “A survey of multimedia and web development techniques and methodology usage”, IEEE Multimedia, 8(1), pp.82-87.

[9] Aun, M. Sh., Yuen, and Agusa, K., (2002), “A framework for debugging client-side reflective and dynamic web applications”, proc. 11th International World Wide Web Conf. WWW2002, Hawaii, USA. http://www2002.org/CDROM/alternate/690/index.html.

[10] Nanard, J. and Nanard, M., (1995), “ Hypertext design environments and the hypertext design process”, Commun. ACM 38(8), pp.49-56.

[11] Turner, C., Fuggetta, A., Lavazza, L. and Wolf, A., (1999), “A conceptual basis for feature engineering”, the Journal of Systems and Software, vol.49, pp.3-15, 1999.

[12] Ousterbout, J., (1998),“Scripting: Higher level programming for the 21st Century”, IEEE Computer, pp. 23-30.

[13] Weblint - Html systax checker, http://filewatcher.org/sec/weblint.html

[14] W3C, “W3C HTML validation services”, available at: http://validator.w3.org.

[15] W3C, “HTML tidy”, available at: http://www.w3.org/People/Raggett/tidy/

[16] Brabrand, C., Moller, A. and Schwartzbach, M., (2001), "Static validation of dynamically generated HTML'', in proceedings of workshop on program analysis for software tools and engineering (PASTE 2001), Snowbird, Utah USA, http://domino.research.ibm.com/confrnc/paste/paste01.nsf.

[17] Atzeni, P., Merialdo, P. and Mecca, G., (2001), "Data-Intensive Web Sites: Design and Maintenance", World Wide Web, 4, 21-47, 2001.

[18] Kienle, H. and Hausi, A. (2001), "Leveraging program analysis for web site reverse Engineering", 3rd Int. Workshop on Web site Evolution, Folrence, Italy.