Pragmatic Reuse in Web Application Development

Josip Maras
University of Split
Faculty of Electrical Engineering,
Mechanical Engineering and Naval Architecture
R. Boskovica 32, 21000 Split, Croatia
josip.maras@fesb.hr

ABSTRACT

Highly interactive web applications that offer user experience and responsiveness of desktop applications are becoming increasingly popular. They are often composed out of visually distinctive user-interface (UI) elements that encapsulate a certain behavior – the so called UI controls. Similar controls are often used in a large number of web pages, and facilitating their reuse would offer considerable benefits. Unfortunately, because of a very short time-to-market, and a fast pace of technology development, preparing controls for reuse is usually not a primary concern. The focus of my research will be to circumvent this limitation by developing a method, and the accompanying tool for supporting web UI control reuse.

Categories and Subject Descriptors

D.2.13 [**Software Engineering**]: Reusable Software

General Terms

Design, Theory, Algorithms, Experimentation

Keywords

Web application, UI controls, code analysis, reuse

1. INTRODUCTION

In the last two decades, web applications have made a tremendous leap forward: from simple static web pages developed only in HTML to complex dynamic web applications developed using server-side technologies, e.g. PHP, ASP.NET, Java that extensively use web services, databases and client-side technologies. Web developers now routinely use sophisticated scripting languages and other active client-side technologies to provide users with rich experiences that approximate the performance of desktop applications [20].

Web application user-interface (UI) is often composed of visually distinctive UI elements, the so called UI controls. Similar controls are often used in different web applications and facilitating their reuse could lead to faster development. Unfortunately, the web application development domain is exposed to a very fast pace of technology development and short time-to-market. This means that preparing code for reuse is often not a primary concern. So, when developers encounter problems that have already been solved in the past, rather then re-inventing the wheel, or spending time componentizing the already available solution (which is sometimes not preferable [7]) they perform reuse tasks [2]. Reusing source code that was not designed in a reusable fashion is known by different synonyms: copy-and-paste reuse [9], code scavenging [8] and, more recently, pragmatic-reuse [4].

Pragmatic reuse treats the system in a white-box fashion and involves extracting functionality from an existing system and reusing it within another system. White-box reuse tasks are complex and error-prone, partly because the goal is to extract the minimum amount of code necessary for the desired functionality [4]. This is particularly true in web development since there is no trivial mapping between source code and the page displayed in the browser; code responsible for the desired functionality is usually scattered between several files, it is often intermixed with code irrelevant for the reuse task and it is written in different languages (PHP, SQL, JavaScript, CSS, HTML) that use different development paradigms.

The overall goal of my PhD research is to increase the efficiency of web UI control reuse by developing a reuse method and the accompanying tool. This is a difficult task because UI controls do not exist as distinct entities in web application code, and because web applications are usually based on dynamic, weakly typed script languages (e.g. JavaScript and PHP) where it is difficult to analyze code dependencies.

2. RESEARCH PROBLEM

My main research problem is: *How to support pragmatic reuse of web UI controls.* This problem can be divided into three research questions:

- How prevalent are UI controls in modern web development?
- How can we locate a subset of the whole web application code and resources that defines the UI control?
- How to enable inclusion of a reused UI control in an already existing web application?

The first question addresses the *usefulness problem* – how much could be gained by supporting web UI control reuse. Even though it is considered that visually and behaviorally self-contained parts of the UI exist, there is no empirical data on how prevalent their use is. This is why it is hard to assess how much effort could be saved if there would exist an automatic method for control reuse.

ICSE '11, May 21–28, 2011, Waikiki, Honolulu, HI, USA
ACM 978-1-4503-0445-0/11/05.

The second question targets the *extraction problem* – from the complete code and resources of the web application, how can we extract only the necessary code and resources. In order to achive this objective, we have to locate code and resources defining the UI control. This is a complex problem because there is no trivial mapping between web application source code and the control shown in the browser. The code responsible for the control is often scattered between several files and is intermixed with code irrelevant for the control; each control can be developed with a number of different languages: HTML, CSS, JavaScript, and PHP. It is important to note that both JavaScript and PHP are weakly typed, dynamic script languages which are hard to analyze and where analysis methods are not that well developed.

The third question addresses the *reuse problem*. Because of the underlying languages, it is hard to reuse the extracted control in an existing web application without introducing bugs due to conflicts (name clashes, function overriding, invalid application of CSS styles, etc.). For this reason a method for detecting and resolving conflicts will have to be developed.

The whole process has to be as automatic as possible, relieving the cognitive load of reuse from the developer.

3. PROPOSED SOLUTION

JavaScript and PHP both include highly dynamic constructs (eval, reflection, functions as variables, etc.) so, information about control-flow, called functions, types of variables, etc. can only be known at runtime. These techniques, while enabling some useful and flexible coding methods, impede static analysis. This is why, inspired by research on typing for dynamic scripting languages [3], I have chosen to base the control extraction and conflict detection on execution trace analysis.

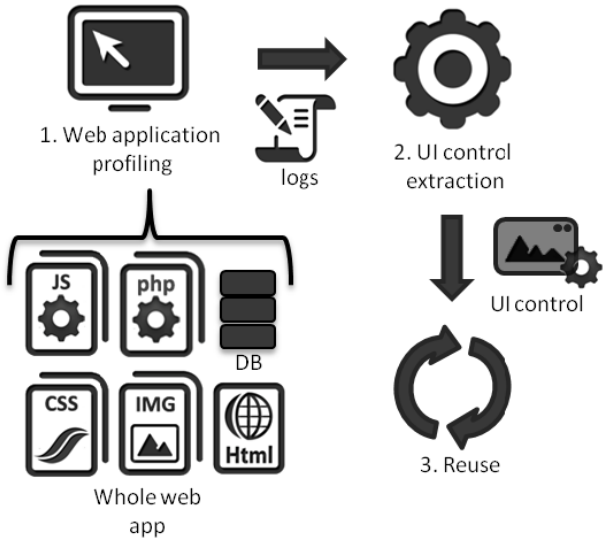

Figure 1: Extracting and reusing UI controls in Web applications

The process is shown in Figure 1 and is divided in three separate steps: 1. web application profiling; 2. control extraction and 3. reuse. The goal of the first step – *web application profiling* – is to derive a set of execution traces that represent the complete behavior of a selected UI control. The best option would be that these traces are derived from UI control tests with high code coverage but, since not all controls have associated tests, we offer the possibility that the developer records a set of interactions which represent the behavior of the UI control. These interactions will then be the basis for generating execution traces. This offers no guarantee that the developer has specified all use cases of the UI control, but at least it is a way to circumvent the lack of proper tests, a situation which is not that rare in web application development. In either case, code responsible for generating execution traces has to be instrumented in order to detect statements that directly modify the user-interface of the control and statement that communicate with the server "from" the control – UI control "significant" statements. This can be achieved by communicating with the browser's debugger to locate these statements on the client-side, and with the server's PHP debugger to locate the statements on the server-side.

In the second phase of *control extraction* the goal is to locate all code and resources necessary for the stand-alone functioning of the UI control. This is done by performing a backward data-flow analysis on the UI control significant statements, aided by the recorded execution-traces. This means that JavaScript and PHP specific data-flow analysis will have to be developed. This is a technically and scientifically challenging phase because a novel dynamic slicing method will have to be developed.

In the third phase – *reuse* – resources extracted in the second phase are embedded into a new, target web application. Naturally, this can lead to various conflicts that have to be detected and handled. Conflicts can occur on different levels: code (JavaScript, PHP, CSS), markup (HTML), and resources (file name clashes) that are all mutually intertwined. While in some cases conflicts can be statically determined (CSS and resources), for the most part (JavaScript, PHP) this problem will also have to be tackled with dynamic analysis. In this case a set of execution traces of the web application that the user control will be a part of will also be required. By providing a dynamic code analysis method that will automatically analyze combinations of execution traces, we can be sure that, by reusing the UI control into a new web application, we haven't introduced any side-effects that will impact the functionality of the rest of the application.

3.1 Contributions

The main contribution of my research will be a method and the accompanying tool for extracting and reusing web UI controls. More specifically, the contributions will include:

- Empirical results on the usage of web UI controls (are there common types of controls, what technologies are used for their realization, how dynamic are they, etc.)
- A code slicing method based on execution traces that will extract code relevant from the selected control perspective
- A code analysis method to detect conflicts between the control and web application in which it will be included
- A tool that performs the extraction, conflict resolution, and control reuse

- Empirical data showing the effectiveness of automatic web UI control reuse

Since JavaScript and PHP are languages based on different paradigms and are responsible for different aspects in web development, code slicing and conflict detection methods will have to be developed and adjusted for each language. They will probably have many similarities, and a more general, dynamic-languages slicing method will possibly be developed.

3.2 Methodology

My research will be based around the following tasks:

- Empirical research in order to determine the current state of web UI controls in web development
- Development of a new approach for UI controls reuse
- Development of a tool for UI controls reuse to validate the usability of the developed technique
- Empirical research on the usefulness of the tool for extracting web UI controls from a random set of Web applications
- A case study to evaluate the usefulness of the technique in real web development projects

3.3 Current state

My first work, dealing with web engineering, was related to modeling of dynamic web applications [13], where I noticed a large number of similar UI controls used in web applications. I followed this with developing a prototype tool, Firecrow [11], for extracting client-side UI controls based on code executed while recording interactions with the UI control (shown in Figure 2). Firecrow is a plugin for the Firefox browser and is available for free [12].

Currently, I am developing a JavaScript slicing method based on the execution traces analysis. I have also done a preliminary empirical study on the usage of UI controls in the client-side web application development domain, by analyzing thirty-five web pages: twenty have been selected from the top 200 most visited web pages in the world [1] (including Google, Facebook, Twitter, and Apple), ten have been randomly selected, and five have been selected from projects in which I have been involved earlier. Although this was by no means a strict empirical experiment, it still offered insight on how common the UI controls are in web application development. On thirty-five web application home pages, 144 web UI controls have been identified - which shows that there is significant potential for reuse. From 144 identified UI controls 133 controls were successfully extracted by Firecrow, while 11 failed [10]. This shows that, even though the advanced JavaScript code analysis is not yet developed, and code that is dead from the UI control perspective is sometimes extracted, the approach has potential to facilitate reuse of a large number of UI controls.

4. EVALUATION

The evaluation of the method and the supporting tool will be based on two approaches. The first is based on extracting UI controls from freely available web applications; the main purpose being to stress test the tool by exposing it to a number of different coding styles and development approaches.

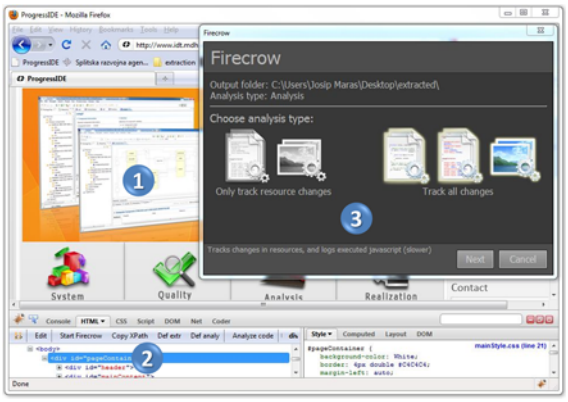

Figure 2: Firecrow user interface – (1) web page chosen for reuse, (2) selecting the user control, (3) Firecrow configuration window.

For this evaluation a categorization of web UI controls and a detailed account on the effectiveness of the approach will have to be made. This should include a percentage of UI controls that were extracted while retaining their behavior and user-experience; and a detailed data on the extracted code (e.g. how much code from the whole application was kept) and resources.

The second approach will be based on a case study application. The main purpose is to evaluate the effectiveness of the approach as a web development method in a real-world web development project. In the near future - I will be a part of a team that will build a commercial web application which will be built from a subset of an already existing web application - iForestFire[15] which was not designed with reuse in mind. During the development, the tool will be used to extract standalone UI controls from the already existing application and integrate them into the newly developed application.

5. RELATED WORK

There exist a number of approaches, environments and tools designed to support reuse. In the web application domain these include HunterGatherer [14], Internet Scrapbook [16], HTMLviewPad [17], and ReWeb [18]; while in the more general domain of reusing Java code there is G&P (Gilligan and Procrustes) [6].

HunterGatherer [14] and Internet Scrapbook [16] allow users to collect components from within Web pages, and to collect components from different Web pages into a newly created page. But since these approaches were developed in 1990's and early 2000, with the term "component" they refer to information components – most usually text paragraphs. These approaches are mostly used to create scrapbooks of data gathered from different web pages, and not to reuse web page controls.

Tanaka et al. [17] describe an interesting approach to clipping and reusing fragments of Web pages in order to compose new applications. They only target HTML elements (specifically HTML forms) and describe how to reroute data entered in a form to original servers that process the request. The applications created in this way are not deployable as standard web applications, but are only executable within their tool – HTMLviewPad. This is the biggest difference

between our approaches: while I aim to facilitate the development of standard web applications by providing a way to reuse already existing code, their approach is more focused on "end-user" programming where users compose new web applications from fragments of old ones, but in a way where newly created applications are only deployable as special applications running inside their tool – HTMLviewPad.

My work is also related to program slicing [19], where by starting from a subset of a program's behavior, the program is reduced to a minimal form which still produces that behavior. My approach can be viewed as web application slicing with the goal of reducing the whole application (along with its code and resources) to a form in which only the visuals, the behavior, and the necessary resources of the selected control are maintained. In the web engineering domain Tonella and Ricca [18] define web application slicing as a process which results in a portion of the web application which exhibits the same behavior as the initial web application in terms of information of interest displayed to the user. In the same work they present a technique for web application slicing in the presence of dynamic code generation where they show how to build a system dependency graph for web applications. This work is dealing mostly with slicing PHP web applications, and was done when web applications were a lot less dynamic on the client side, and when OO design and AJAX applications were not yet widely spread. While still being relevant, a lot has changed in the web development practices and this algorithm needs improvements.

In the more general domain of Java applications, G&P [6] is a reuse environment composed of two tools: Gilligan and Procrustes, that facilitates pragmatic reuse tasks. Gilligan allows the developer to investigate dependencies from a desired functionality and to construct a plan about their reuse, while Procrustes automatically extracts the relevant code from the originating system, transforms it to minimize the compilation errors and inserts it into the developer's system. This work was further expanded [5] with the possibility to automatically recommend elements to be reused based on their structural relevance and cost-of-reuse.

6. ACKNOWLEDGMENTS

This PhD research is partially supported by the Swedish Foundation for Strategic Research via the strategic research center PROGRESS, and the Unity Through Knowledge Fund supported by the Croatian Government and the World Bank via the DICES project. I would also like to thank my advisors: Maja Štula, Jan Carlson and Ivica Crnković for steering my PhD topic and for many helpful comments on this paper.

7. REFERENCES

[1] Alexa. Alexa top sites, October 2010. "http://www.alexa.com/topsites/".

[2] J. Brandt, P. J. Guo, J. Lewenstein, and S. R. Klemmer. Opportunistic programming: How rapid ideation and prototyping occur in practice. In *WEUSE '08: Workshop on End-user software engineering*, pages 1–5. ACM, 2008.

[3] M. Furr, J.-h. An, J. Foster, and M. Hicks. Profle-Guided Static Typing for Dynamic Scripting Languages. In *24th Annual Conference on Object-Oriented Programming Systems, Languages, and Applications*, 2009.

[4] R. Holmes. *Pragmatic Software Reuse*. PhD thesis, University of Calgary, Canada, 2008.

[5] R. Holmes, T. Ratchford, M. Robillard, and R. J. Walker. Automatically Recommending Triage Decisions for Pragmatic Reuse Tasks. In *ASE '09: Proceedings of the 2009 24th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, 2009.

[6] R. Holmes and R. J. Walker. Semi-Automating Pragmatic Reuse Tasks. In *ASE '08: Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, pages 481–482. IEEE Computer Society, 2008.

[7] C. Kapser and M. W. Godfrey. "Cloning Considered Harmful" Considered Harmful. In *WCRE '06: Proceedings of the 13th Working Conference on Reverse Engineering*, pages 19–28. IEEE Computer Society, 2006.

[8] C. W. Krueger. Software reuse. *ACM Comput. Surv.*, 24(2):131–183, 1992.

[9] B. M. Lange and T. G. Moher. Some strategies of reuse in an object-oriented programming environment. *SIGCHI Bull.*, 20(SI):69–73.

[10] J. Maras. Firecrow evaluation, December 2010. "http://www.fesb.hr/~jomaras/?id=apps-examples".

[11] J. Maras. Home page of the Firecrow tool, December 2010. "http://www.fesb.hr/~jomaras/?id=app-Firecrow".

[12] J. Maras, M. Štula, and J. Carlson. Extracting Client-side Web User Interface Controls. In *ICWE 2010, International Conference on Web Engineering*, pages 502–505, 2010.

[13] J. Maras, M. Štula, and I. Crnković. phpmodeler - a Web Model Extractor. In *ASE '09: Proceedings of the 2009 24th IEEE/ACM International Conference on Automated Software Engineering*, pages 660–661, 2009.

[14] M. Schraefel, Y. Zhu, D. Modjeska, D. Wigdor, and S. Zhao. Hunter Gatherer: Interaction Support for the Creation and Management of Within-Web-Page Collections. In *11th international conference on World Wide Web*, pages 172–181, 2002.

[15] D. Stipanicev and M. Stula. iForestFire, December 2010. "http://ipnas.fesb.hr/index.php?lang=en".

[16] A. Sugiura and Y. Koseki. Internet scrapbook: creating personalized world wide web pages. In *CHI '97: Extended abstracts on Human factors in computing systems*, pages 343–344. ACM, 1997.

[17] Y. Tanaka, K. Ito, and J. Fujima. Meme Media for Clipping and Combining Web Resources. *World Wide Web*, 9:117–142, 2006.

[18] P. Tonella and F. Ricca. Web Application Slicing in Presence of Dynamic Code Generation. *Automated Software Engg.*, 12(2):259–288, 2005.

[19] M. Weiser. Program slicing. In *ICSE '81: 5th International Conference on Software engineering*, pages 439–449. IEEE Press, 1981.

[20] A. Wright. Ready for a Web OS? *Commun. ACM*, 52(12):16–17, 2009.