Gathering Data for Contemporary Canonical Software Courses

Ritika Jhangiani
Department of Computer Science
College of St. Scholastica
1200 Kenwood Ave., Duluth, MN 55811
rjhangia@css.edu

Elizabeth Ellis
Department of Computer Science
Hiram College
11694 Hayden St., Hiram, OH 44234
ellisea@my.hiram.edu

Advisor: Yuanfang Cai
Department of Computer Science
Drexel University
3141 Chestnut St., Philadelphia, PA 19104
yfcai@cs.drexel.edu

***Abstract*—In an effort to offer contemporary canonical software courses in web browser and web server development, it is necessary to understand the key technologies behind designing a web server and a web browser and how a web server and a web browser interact with each other. To do this, we studied Apache HTTPD and Mozilla Firefox to understand their architecture structure and how they survive over years, including their development history and developer interaction. This paper presents our initial work in organizing knowledge to present in the newly offered courses.**

I. Introduction

Computer science curricula usually include the teaching of canonical software systems, such as operating systems and compilers, because these systems embody the concepts and technologies that constitute an important part of the knowledge base of computer science education. The proliferation of the Internet has changed the way people interact with computers. Hardly a day goes by without using a web browser, checking email, or instant messaging. The concepts and technologies behind these new canonical systems are important computer science knowledge. In particular, understanding how web browsers and web servers function and interact is fundamental to Internet-based application development. It becomes important for students to be exposed to the formal study of these modern canonical systems.

Our task was to mine the source code, developer mailing lists, and SVN/CVS commit log information for both the Apache HTTPD and Mozilla Firefox projects. To do this, we had to reverse-engineer the software.

The difficulty in doing these things is that these projects haven't been studied in such a close proximity before, and those that did studied them at a high level which would not be conducive to the

knowledge needed to offer courses in web browser and web server development. Another reason that these projects were hard to study is that they are very large, which makes most common reverse-engineering tools impossible to use as they cannot handle so many source files at once.

This paper explores our preliminary work in organizing the data necessary to study these projects closely. We have gathered pertinent information from the developer mailing list of the Apache HTTPD and the CVS commit information for Mozilla Firefox into a database using scripts written in Ruby. Finally, we are currently working on reverse-engineering the Apache source code using the doxygen open-source tool to gain dependency information to use in creating dependency structure matrices (DSMs) in order to understand Apache's architecture.

II. Related Work

Many studies have been conducted previously on the Apache HTTPD server and Mozilla Firefox. For instance, Dragoi and Preston have studied the concrete architecture of the Apache server and found it to consist of the main core and many individual modules [1]. Campos et. al studied the architecture of Mozilla Firefox and found it to be a match for a layered architectural model, with the main components being independent as long as the component conforms to a predefined interface [2].

However, the authors do not go into the technical details about the interaction of web servers and web browsers or into the low-level details of the architecture and software design.

III. Gathering Data

The goal of the project we were a part of was to develop new courses to offer students in the 2011-2012 school year. These courses would focus on the design and interaction of web servers and web browsers. In order to offer such courses, an intricate and in-depth level of knowledge on these types of software is necessary.

Our task was to gather that knowledge from two popular open source software projects–the Apache HTTPD web server and the Mozilla Firefox web browser. These projects have been successful and have survived for many years, making them feasible for study as their source code and development history is made public.

A. Source Code Repositories

To understand the development history of these software projects, we had to mine the source code repositories. This is necessary to understand how the structures and features evolve over time, where and when refactoring happened, and which and how many other parts of the system changed accordingly.

To do this, we downloaded the Mozilla Firefox CVS commit log. We extracted pertinent information such as the author of the commit, the date, the time, the revision number, the bug number (if applicable), and comments using Ruby scripts. We organized this information and stored it in a MySQL database.

B. Developer Mailing Lists

It is also crucial to mine the developer mailing lists for both projects. Knowledge gained from developer communications includes how the number of developers change over time, how many sub-communities there are, and how the communications correspond to modules.

To do this, we downloaded the Apache HTTPD developer mailing list from their web site using wget on a Linux server. We extracted pertinent information from each e-mail such as the sender, the date, the time, the subject, and the original e-mail this e-mail is responding to (if applicable) using Ruby scripts. Using this information, we derived communications between two developers as people. This will be used in the analysis of the information. We organized this information and stored it in a MySQL database.

C. Reverse-Engineering the Source Code

This step was met with some difficulty. It is probably the most important piece of the puzzle in understanding the architecture and design of these two systems.

Many tools (such as REportal, Bunch, and Lattix) would not work for us in extracting the main architectural model of these two systems [3][4]. Some common reasons that existing tools prove unsuccessful in our research are: the tool doesn't have the language compatability with C/C++ that Apache and Firefox are written in, the large size of the source code of these projects, and that the tools were still under development.

However, we were able to successfully gain information from the doxygen documentation tool for the Apache HTTPD server version 2.2 in the form of XML and HTML output containing dependency call graph information. By writing XSLT stylesheets, we extracted relevant dependency information between modules, functions, and classes. This will be used to analyze the interaction of the modules, functions, and classes to gain an understanding of the lower-level architecture of the Apache HTTPD web server.

IV. Future Work

Although we have made excellent progress, we have not yet compiled data for vital parts of both projects. We hope to extract the Apache SVN commit log information using a Perl script written by a graduate student at Drexel University. In the past we have been unable to download the Mozilla Firefox developer mailing list as the permissions were forbidden, but we hope to gain the data from the mailing list in the future using Ruby scripts.

We also hope to reconfigure the doxygen files using XSLT to get specific dependency information that we will use as input to the uml2acn and Minos tools to generate augmented constraint networks (ACNs) and dependency structure matrices (DSMs) [5].

By doing this, we will understand the lower-level architecture of the two systems, making it possible to develop a teaching tool to offer courses in web server and web browser development to incoming students.

V. Acknowledgements

We would like to thank Drexel University's Software Engineering Research Group and the CRA-W for their help and support on this project.

References

[1] O. A. Dragoi and J. E. Preston, "The concrete architecture of the apache web server," http://www.cs.ucsb.edu/ tve/cs290i-sp01/papers/Concrete_Apache_Arch.htm, Feb. 1999.

[2] A. C. et. al, "Conceptual architecture of firefox," http://web.uvic.ca/ hitchner/assign1.pdf, 2007.

[3] S. Mancoridis, T. Souder, Y.-F. Chen, E. Gansner, and J. Korn, "Reportal: a web-based portal site for reverse engineering," in *Reverse Engineering, 2001. Proceedings. Eighth Working Conference on*, 2001, pp. 221–230.

[4] S. Mancoridis, B. Mitchell, Y. Chen, and E. Gansner, "Bunch: A clustering tool for the recovery and maintenance of software system structures," *Software Maintenance, IEEE International Conference on*, vol. 0, p. 50, 1999.

[5] S. Huynh, Y. Cai, and W. Shen, "Automatic transformation of uml models into analytical decision models," Drexel University, Tech. Rep. DU-CS-08-01, Jan. 2008, https://www.cs.drexel.edu/files/ts467/DU-CS-08-01.pdf.