Creating Synergy Between Usability Courses and Open Source Software Projects

Daryl H. Hepting, Lijuan Peng, Timothy J. Maciag, David Gerhard, Brien Maguire
Computer Science Department
University of Regina
Regina, SK S4S 0A2 CANADA
{dhh,peng200l,maciagt,gerhard,rbm}@cs.uregina.ca

Keywords: Usability, instruction, open source software

Abstract

In this paper, we discuss our experience in offering a usability course with projects taken from an active open source software development project. We describe what was done in the class inside the larger context of the usability of open source software. We conclude with an invitation for others to adopt this model and use it for their own purposes.

1. Introduction

A common model for our offering of our *Human Computer Communications* course, intended to cover HC1-HC5 of the Computing Curricula 2001[3], has been to put students together in groups and have them design an interface to a contrived system of their choosing. On the whole, this approach has been disappointing, with only a few happy exceptions. A major difficulty is that students have not seen the importance of proper design for the user interface.

For the past several terms, Rosson and Carroll's *Usability Engineering*[11] has been used as the main text for the course. The focus on plain language scenarios has helped students to focus on user interface design and to understand the rationale for a focus on interface design. However, the lack of an existing software application to which the students' knowledge could be applied seemed to lessen the impact of the message.

Therefore, two key modifications were made before the Fall 2006 offering of the course. The first change was to base the coursework on an active open source software development project. Our department has a long tradition in open source software and the association with an active open source project made the course more practical and responsive to current trends. The second change was to move the discussion of evaluation methods from the latter part of the course to the beginning. As McConnell[8] highlights, most of a programmer's life will be spent maintaining code written by others. It seems true also that interface designers will be most commonly asked to improve existing graphical user interfaces (GUI) or add GUIs to existing command line systems. This change was intended to be disruptive for the students, so that they might less complacent about the design process.

The topic of open source usability has gained much prominence recently [13,10]. Open source project communities may have the concern of outsiders engaging in "drive-by" usability, defined by Sinha[13] as an: "expert who does not understand their project or goals and gives them a bunch of usability suggestions after a quick review." Students in the course were new to usability, but they did spend considerable time with the application and the evaluation results that were available. We felt that having the students approach a project with fresh eyes could lead to valuable insights for the project.

The framework described here will continue to be developed. It is our hope that members of the community will accept the invitation to build on what has been started here.

2. Background

Nichols and Twidale [2] proposed a number of approaches to problems of open source usability, including technological, commercial, and academic approaches. They also describe the possible roles of experts in the process, although they do not address the potential hostility from the community observed by Sinha[13].

Scacchi [12] focused on the requirements process and completed an empirical study of four open source communities, exploring the development of requirements by and for the community.

Zhao and Deek[14] applied "exploratory learning" (learning with the aim of completing the current task only) to help common and new users contribute more to open source usability. HCI patterns are used as a guide for usability inspection, and users find usability problems while learning usability knowledge.

Carroll and Rosson [1] employ specific cases and scenarios in their usability education practice.

Besides researchers, there are projects organized for improving the usability of open source software. OpenUsability.org[10] "…brings Open Source Developers and Usability Experts together". There have been more than one hundred projects and more than one thousand registered users at this site. It seems that communities who register their projects there are willing to have experts perform "drive-by" usability improvements.

3. New Course Framework

We found that working with an open source software development project gave students an excellent experience and a real grasp of the importance of interface design issues. Although this course offering did not deal with the

project's code repository, the knowledge that students were working on an active project was important to their motivation and sense of accomplishment. The following outlines the organization of our new course framework.

3.1 Project Selection

Prior to the start of the semester, an active open source project was identified. The Department of Computer Science at the University of Regina operates an Open Systems Lab as a focal point for open source software development activities. Both research and teaching in the department benefit from a synergistic view of software. In particular, open source software developed in one class or research project could be used in other classes and research projects.

The first candidate was Nvu, which was "started from the Mozilla Composer code base. The Mozilla Internet suite is in the process of being broken up into individual pieces (browser, email, etc.). Nvu will pick up where Composer left off, adding additional features, functionality and ease of use."[9] This web development application would have fit well with an existing departmental strength in multimedia. Because of the course focus, project activity was assessed by the amount of traffic on the project's e-mail lists, so Nvu was dropped from consideration because of very low e-mail traffic.

The second candidate was LyX, which produces output using LaTeX. Because of our familiarity with LaTeX and because of its very active e-mail lists, the LyX[7] project was chosen. "LyX is the first WYSIWYM [What You See is What You Mean] document processor. LyX is a document processor that encourages an approach to writing based on the *structure* of your documents, not their appearance."

3.2 E-mail list survey

A survey was sent via e-mail to the project's user and developer e-mail lists immediately before the start of the semester. The goal was to have the people actively involved in the project (as user or developer) to assess the features of the software that were identified on the project website. Respondents were asked to compare these features to other similar software that they had previously used (comparisons were made with Microsoft Word, Open Office, Corel WordPerfect, and others). Developers were also asked about their vision for the software. Both users and developers were asked to make additional comments about the project. The intent of this survey was to gain a sense, from those close to the project, of how well the software matched up with its stated goals.

Students were given this anonymized survey data to instructions to be perform a simple statistical analysis as part of a laboratory exercise. This was first point where students came to understand the opportunities for design.

3.3 User study

Students gained experience in developing a user study and running it. The study was inspired by the comparisons that naturally occurred between lyx and Microsoft Word (as an application that was most widely familiar).

Each participant was responsible for completing tasks in both lyx and Microsoft Word. The evaluation provided useful insight into ideas for new designs. Participants were recruited from the University of Regina Computer Science Participant Pool[6], which gives students in undergraduate students in computer science courses the opportunity to earn research credits (bonus marks) for their participation in departmental research. A student may apply a maximum of 2 credits to any one class. The study allowed the students to gather more information about the usability issues in LyX and to gain important understanding of how to think from the point of view of users.

3.4 Archive analysis

Important sources of information in terms of requirements gathering were: e-mail archives for the user group; requested features in the bugzilla; and the feature poll on the project wiki.

The web-based information is convenient for studying the current practices and finding opportunities for design. Most students did not have any experience with LyX before the class and they learned about it through their own experimentation with it. The archive analysis was a good way to collect the requirements of LyX, but analysis was required to consolidate the information dispersed within the different sources[12].

3.5 Theme Identification

From the students' analysis of the e-mail surveys, the user study data, and the archive contents, each student group developed a list of themes[11] for issues uncovered in the project. Later, each student group selected a theme as the basis for developing new ideas to incorporate into the design of the Lyx GUI. Rather than have each group consider the whole GUI, the student groups were asked to consider one issue in some depth, which was chosen in an attempt to get the biggest bang[5] in terms of the GUI's improvement. The particular themes and issues identified in this process are not discussed in this paper, but may be found online at:
http://www.cs.uregina.ca/~hepting/teach/openhci/.

3.6 Scenario-based Design

Once the themes were identified, each group proceeded to develop problem, activity, information design, and interaction scenarios[11]. These scenarios, accompanied by claims analyses, which highlighted the positive and negative aspects of the scenario features, helped the groups to refine their proposed solutions. The scenario-based approach did help students to communicate easily, and they

benefited from the ability to iterate and improve their solutions.

3.7 Scenario Machines

The interface design solutions were prototyped in Microsoft .net because of its familiarity and availability (through the MSDNAA program). The intent was to present a user with the impression of a fully functional system while really only providing functionality corresponding to the particular issue addressed. Because of its proprietary nature, an open source solution would have been preferred and one will be investigated for future offerings of the class. However, because students are using this platform only for prototyping, the choice of tool is ultimately less important.

3.8 Evaluation

Students in the class evaluated the scenario machines from other groups. This gave the students another opportunity to assume the roles of experimenter and user (each student played both roles at different times). At earlier times in the course, students were asked to critique the work products of another group. This feedback also helped students to prepare a better final product.

4. Discussion and Future Work

The end result of the semester's work was satifisying, but there are some areas that need to be addressed.

Students in the class appreciated the involvement with an open source project, even though they did not actually work with the project code repository. They did appreciate the fact that their work might be incorporated into a later version of the LyX software. The work undertaken in this class was intended to be a real contribution to the LyX project, however there was not sufficient infrastructure in place to allow the easy transfer of the course work back into the project. The choice of focus on design over coding would be made again, but students need to be more connected to the selected project and in conversations with its users and developers during the course.

Plans are now underway to create a dedicated site for this framework that would include a library of resources from past projects, inspired by Carroll and Rosson[1], and the support to capture all aspects of new usability projects. This site would include technology, such as a wiki, that could help to keep the open source software project developers in touch with usability project students over the duration of the course, and enable them to take insights back into their development.

The students found the course work to be beneficial for their understanding of the course concepts. Regrets were expressed that they did not work on a project that they were more likely to use again on a regular basis, since LyX is not aimed at the general word processor user.

Projects might be selected according to a different metric[4] in future, to better match the interests of students.Some consideration is being given to allowing student groups to choose their own open source software project with which to work. In this case, the students would be responsible for designing and administering the initial e-mail survey, which would necessitate some changes to the course timing. In this vein, more evaluation material could have been presented during the initial phase of the class, to the benefit of later activities.

Although student groups dealt with individual issues, some additional coordination between groups in order to improve the overall consistency between all the design improvements. As it was, each group presented their own particular design in a vacuum. Relating this design, in terms of feasibility perhaps, to the existing code base and direction of the project would have also been helpful.

Direct contributions to the code repository were not made in this course, but this has led to the offering of a standalone course on Open Source Software Development which may prove to be an important, if not essential, first step for successful adoption of open source projects in other classes. Students from this usability class (CS305) took the open source development course (CS490CN) and enjoyed the experience. As CS490CN becomes more established, and more CS305 students take it as a follow-up course, there will be distinct opportunities to create more synergy between the two courses: interface design completed in CS305 could be turned into patches submitted during CS490CN.

Acknowledgements

We thank the LyX community for their interest and involvement, the Department of Computer Science at the University of Regina, and the students of CS305 in the fall semester of 2006, in particular: Thomas Boxall, Kevin Cannon, and Michael Hamilton. Brad Henry, instructor for the open source software development course, also provided valuable insights.

References

[1] Carroll, J. M. and Rosson, M. B. A Case Library for Teaching Usability Engineering: Design Rationale, Development, and Classroom Experience, ACM Journal on Educational Resources in Computing, Vol.5, No.1, 2005, pages 1-22.

[2] Nichols, D. M. and Twidale, M. B. The Usability of Open Source Software, *First Monday*, Volume 8, Number 1, 2003, http://www.firstmonday.org/issues/issue8_1/nichols/, Accessed February 15, 2008.

[3] CC2001, "Computing Curricula 2001 (Web Site)," ACM and IEEE-CS, http://www.sigcse.org/cc2001/, Accessed February 15, 2008.

[4] Fogel, K. *Producing Open Source Software,* O'Reilly, 2005.

[5] Gilb, T. *Principles Of Software Engineering Management,* Addison-Wesley, 1988.

[6] Hepting, D. Ethics and usability testing in computer science education, SIGCSE Bulletin, 38(2), 2006, pages 76-80.

[7] Lyx, www.lyx.org, Accessed February 15, 2008.

[8] McConnell, S. *Code Complete*, 2nd Edition, Microsoft Press, 2004.

[9] Nvu, http://nvudev.com/about.php, Accessed February 15, 2008.

[10] OpenUsabiity, http://openusability.org, Accessed February 15, 2008.

[11] Rosson, M. B and Carroll, J. M. Usability Engineering: scenario-based development of human-computer interaction. Morgan Kaufmann, 2002.

[12] Scacchi, W. Understanding the Requirements for Developing Open Source Software Systems, IEE Proceedings Software, 149(1), 2002, pages 24-39.

[13] Sinha, R. Open Source Usability: The birth of a movement, http://www.rashmisinha.com/archives/05_04/open-source.html, Accessed February 15, 2008.

[14] Zhao, L. and Deek, F. P. Exploratory Inspection – A Learning Model for Improving Open Source Software Usability, CHI 2006 extended abstracts on Human factors in computing systems, 2006, pages 1589-1594.