Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2005 Proceedings

Americas Conference on Information Systems (AMCIS)

2005

Improving Open Source Software Usability

Luyin Zhao
Rutgers University, luyin@pegasus.rutgers.edu

Fadi P. Deek
New Jersey Institute of Technology, fadi.deek@njit.edu

Improving Open Source Software Usability

Luyin Zhao
Rutgers University
luyin@pegasus.rutgers.edu

Fadi P. Deek
New Jersey Institute of Technology
fadi.deek@njit.edu

ABSTRACT

Usability of open source systems is becoming a relevant topic for investigation given the proliferation of open source software. General assumptions do not favor a positive relationship between the existing open source development model and good usability due to a host of social context reasons. In this paper we provide empirical evidence on this issue through a case study using an open source project called Ganttproject. Results point to likely deficiencies in the open source model. An analysis of this phenomena and some potential solutions for improving open source usability are proposed.

Keywords

Open source software, usability, case study, protocol analysis, usability guideline, usability testing, heuristics.

INTRODUCTION

The unique open source development model features massive distributed collaboration, peer review, incremental development, and evolution through quick version release. User participation is the cornerstone of open source success, and is also where the distinction between users and developers becomes blurred. Despite the fact that numerous studies have examined the evolution of open source software from a functionality point of view, there has been little discussion on open source usability, especially as it relates to whether user contributions actually help to improve open source usability.

The literature suggests that there is little evidence that OSS projects devote much resources to usability improvement, or a demonstration of a strong relationship between the unique OSS development model and better software usability (Nichols and Twidale, 2003; Twidale and Nichols, 2004). The arguments tend to revolve around the idea that usability is an art form that requires a dedicated effort from a usability expert (or a team of experts). This argument is based on two premises: 1) usability experts don't "do" open source projects, and 2) dedicated efforts are impossible for the open source development paradigm (Salmoni, 2004) (Benson, Muller-Prove et al., 2004). While these assertions are based on current observations of OSS, there is still a great potential for the open source movement to address usability. We offer some reasons for this statement: OSS projects have begun to attract usability engineers; there is a great flexibility for usability professionals to contribute; there is a large user base for usability testing; and it is possible to build automatic bug reporting features into software (Salmoni, 2004). This paper provides further insights into this issue and offers in-depth discussions to support our viewpoints through the use of a case study.

A SOCIAL CONTEXT ANALYSIS

One of the intangible motivations to participate in OSS development is self-satisfaction and desire to enhance one's reputation among community members. The implication for usability is that "when programmers produce open source software, since they are largely scratching their own itch, they tend to produce software for themselves, and in particular be perfectly content with the (programmer-oriented) user interface" (Pemberton, 2004).

The social context of open-source software development does not include the typical user and so it may be unreasonable to expect their interests to be represented in the programs, not to mention specific efforts to accommodate human diversity required by fine usability engineering as described by (Shneiderman, 1998; Messerschmitt, 2004) suggests that software (usability) defects are typically misunderstandings of users' need and "opening source code" does not help. He argues that for naïve users, testers, program managers, marketers, salespeople and customer service play a crucial role in understanding their needs. Because these users typically are unable to participate in OSS development, therefore most OSS lack usability from the perspective of naïve users.

Furthermore, 'hard' algorithmic problems have a greater value in the 'reputation market' than issues of usability (Nichols, Thomson et al., 2002; Twidale and Nichols, 2004) found that similar to traditional software development, it is difficult for usability advocates to establish legitimacy arguments against countervailing expert-user functionality-centric claims. These

assumptions have been validated through several usability evaluations of OSS projects. (Nichols, Thomson et al., 2002) evaluated the OSS project Greenstone and identified four design issues that generated these usability problems:

- Developer-user knowledge gap
- Developer-user working context difference
- Developer-user interaction style gap
- Unclear and misused documentation

(Benson, Muller-Prove et al., 2004) identifies the following problems with the GNOME project despite the efforts and resources devoted to the usability aspects of this effort:

• Communication: Non-technical users might be intimidated by dedicated bug reporting channels that require complicated procedure for registration and knowledge for accessing complex bug databases.

• Early input: Efforts for usability improvement usually come after the interface have been built. Compare to software functionality that benefits from releasing source code early and frequently in order to get feedback, usability might suffer from this type of "later release".

• User profiles: There is no shared or documented vision of GNOME'S target audience.

• Process: No obvious usability methodology is employed. Many developers contribute during their spare time, so they are understandably reluctant to follow heavyweight processes.

In general, it is fair to assume that the social context of OSS development has set a number of barriers to usability improvement. That also explains why many successful OSS projects are operating systems and compiler types that require little human-computer interaction.

A CASE STUDY

To validate the assumptions put forth earlier, we conducted a usability evaluation case study using an open source software called Ganttproject. Gantt chart, developed by Henry Laurence Gantt, is a useful and well-known project management tool that uses graphical representation to display and manage tasks against the progression of time. Ganttproject is an open source software that offers computerized Gantt chart creation and maintenance capability. Launched in 2003, this project has evolved into a popular tool in the open source community. The current version is 1.10.3. It is available for free download at http://ganttproject.sourceforge.net. We chose Ganttproject not only because it is a GUI application, but also because it was a work-in-progress at version 1.9.8 so that we would be able to assess how usability problems were raised and addressed after one year as the software becomes more mature.

Protocol analysis

A "protocol" is a record of a step-by-step procedure. Protocol Analysis using the think aloud method represents an effort to uncover the thinking process of a software system user in order to analyze aspects of the software that contributes to its usability as well as the elements that detracts from its usability. In this method, one records the step-by-step procedures of a user "thinking out loud" while trying to use a certain computer system (Boren and Ramey, 2000) By encouraging the user to think out loud through the course of a series of tasks one can gain insight into the difficulties the user experiences in making the system perform its tasks and conform to the user's expectations of how the system should respond to inputs. To the extent possible, the researcher performing the analysis is removed from the situation except for providing, as needed, encouragements to the subjects to continue to describe their activities. If the protocol analysis is conducted successfully, the transcript of the session will be a report on the usability of the system tested.

Our first task was to find out usability problems of Ganttproject version 1.9.8. To do this, we conducted a protocol analysis using Ganttproject version 1.9.8. Six human subjects with different backgrounds and expertise were asked to go through five typical tasks that cover the major functionalities of the tool. The detailed steps of the protocol analysis are summarized in table 1. The protocol analysis yielded ten significant usability problems. These problems are documented in Appendix 1.

Steps	Description
Software introduction	Briefly introduce Ganttproject software to the subject and show a flash animation of typical usage of the software.
Experiment introduction	Explain the purpose of the experiment, the tasks to be performed by the subject, and the role of the observer.
Sign consent form	The subject is asked to sign a consent form to agree to participate in the experiment.
Conduct protocol analysis	The subject goes through the five tasks while "thinking-aloud". All subject verbalizations during task execution are recorded for later analysis. For the observer, we followed the guideline suggested by (Someren et al., 1994) to get least observer interference and maximum subject output.
Filling out questionnaire	A questionnaire about the software interface is filled out by the subject
Protocol analysis	The recorded protocol is reviewed and analyzed to uncover usability problems encountered during the subject's task execution process.

Table 1 – Protocol analysis steps

A retrospective study on usability improvement

The current version of Ganttproject is 1.10.3 as of December 8, 2004. It has been over a year since our protocol analysis was conducted. We have subsequently reexamined whether the usability issues identified in our project are addressed by the latest version.

As an open source project, Ganttproject uses the standard bug reporting and feature request mechanisms provided by sourceforge.net (the largest open source hosting site worldwide). Any user feedback could be submitted to both repositories. Since the launch of the project, there have been 168 bugs and 149 feature requests submitted to the repository. For the 149 feature requests, 108 are still open (not resolved yet) and 41 are closed. For our study, whether these bugs and feature requests are classified as usability-relevant issue is based on the Jacob Nielsen's usability heuristics (http://www.useit.com/papers/heuristic/heuristic_list.html).

Table 2 shows how usability problems found by our protocol analysis and by other Ganttproject users are distributed among different usability categories. The symbol "*" represents one usability issue. In the bug repository, only three of the bugs are related to usability. As for feature requests, 37 of those are usability relevant.

Heuristics	Think aloud	Bug reporting	Feature requests (closed)	Feature requests (open)
Visibility of system status	A7		**	******
Match between system and the real world			***	
User control and freedom	A1, A4			*
Consistency and standards	A4, A10			*
Error prevention	A3, A5, A8			*
Recognition rather than recall	A6, A7		*	*
Flexibility and efficiency of use	A2		*********	*******
Aesthetic and minimalist design	A9	**	***	**
Help users recognize, diagnose, and recover from errors				
Help and documentation		*		

Table 2 – Usability problems found

Findings

- Our protocol analysis method discovered ten usability problems. Over a one-year period, only four of them (A2, A5, A6, A10) have been resolved and implemented in the latest version. Only two of the ten usability problems (A1 and A3) were also identified and reported by other users.

- The bug reporting mechanism of open source apparently is not considered as an appropriate channel for reporting usability problems by users. An analysis of the bug repository only finds three usability relevant problems.

- Feature requests form a large repository for usability enhancement suggestions. However, most of the requests are submitted by advanced users as they frequently look for new features that enable "flexibility and efficiency of use" and "visibility of system status", which are only two aspects of usability heuristics.

- It is evident that if no formal usability testing (such as think aloud) is performed, open source projects would still lack the effective mechanisms to discover usability problems, especially those encountered by typical or non-experienced users, such as error prevention. This issue might be alleviated by explicitly establishing channels to remedy this such as "usability bug reporting", or more advanced web-based usability testing methods.

- Typical open source projects do not have sufficient resources to address usability problems and enhancement requests. For example, one response for a request to add "undo" feature to the software is that "…Yes I think, after reformatting the soft... Because for the moment it's a quit difficult to implements it...".

MECHANISMS FOR IMPROVING OPEN SOURCE USABILITY

The results of our case study reveal potential deficiencies in the current open source development model as it relates to usability, especially when the end-users are not computer professionals. In light of this problem, researchers and practitioners have proposed a number of mechanisms including:

- Instituting formalized procedures to enforce usability guidelines might be difficult in OSS projects unless project leaders are able to enforce control and direct resources to support such efforts. For instance, the well known GNOME and KDE projects have published Human Interface Guidelines to help developers write applications that are consistent with the GNOME and KDE environment, although how the guidelines are actually followed is unknown. Even though applying guidelines seems to be a feasible approach, it might be a problem with OSS since it tries to bring more formalized process and style into OSS development, thus violating the "OSS philosophy". Furthermore, certain obvious drawbacks of guidelines are they can range from general recommendations to specific style guide (Jeffries, Miller et al., 1991), they are usually maintained as prescriptive checklists without context and examples, and only a small portion of guidelines are kept in designers' working memory.

- Large corporations to contribute to the open source projects using their expertise and resource of usability engineering (Benson, Muller-Prove et al., 2004; Nichols, McKay et al., 2003). More importantly, such commercial support and control would be able to not only direct more resources required to improve usability, but also could bring usability experts into OSS projects. One example is that, from March 2001, Sun has conducted formal usability testing and established usability community for GNOME project.

- Automated usability testing is another approach to reduce cost, increase consistency and reduce human involvement, which is a limitation of OSS projects (Nichols, McKay et al., 2003; Salmoni, 2004). Automated testing requires less resource. However, due to the fact that OSS development teams do not even invest sufficient time in conducting functionality testing because they believe peer review and user contribution would find the bugs (Zhao and Elbaum, 2003), and thus whether usability testing would get enough attention from development team is still questionable.

- From a post-deployment reporting perspective, (Nichols, McKay et al., 2003) proposes to improve the design of bug reporting capability of OSS in order to take advantage of the large user base, especially non-active users that are geographically spread. This seems to be the most promising way to improve OSS usability because it opens up a new channel for the typical users to contribute. Not only automated methods for extracting usability information can be used (Hilbert and Redmiles, 2000), but also a well-designed bug reporting process would assist in capturing more cognitive information from typical users. Since experienced users are able to find and fix so many functionality bugs, why can't non-experienced (more typical) users find usability bugs? The key success factor for this method is that a good communication channels must be established for non-expert users to contribute because typically, as we noted before, these users can be intimidated by the complicated procedure to record bugs into repository. One example, called critical incident reporting, is to embed bug reporting mechanism into the software so that both system state information and user subjective comments can be transmitted back during system usage. The screenshot from an open source software "more.groupware" (shown in Figure 1) illustrates this method.

Figure 1 – Bug reporting embedded in the software

Despite the fact that methods like critical incident reporting opens up new channels for typical users to contribute, there remains a problem that limits the effectiveness of such approaches: unlike usability experts, typical users don't have sufficient knowledge about usability. Unfortunately, the end-users have usually, and perhaps subconsciously, blamed themselves for difficulties with awkward user interfaces (McCoy, 2002). Therefore, research efforts should also focus on how to train end-users or provide effective assistants for them to inspect the software quickly discover usability problems. Such knowledge transfer should not be based on long-term training but on quick context-based usability knowledge base such as usability patterns (van Welie and van der Veer, 2003).

CONCLUSION AND FUTURE WORK

The results of our case study point to the weak tie between open source development model and usability. First, the community-based development model is yet to establish an appropriate channel for reporting usability problems. Second, most of the contributing users tend to be experienced ones. Their requests for usability improvement do not necessary reflect those requirements of typical or non-experienced users.

Under the circumstance of lacking resources to conduct formal usability evaluations, getting the typical users to become involved through remote usability evaluation seems to be the most viable approach. Further research in this direction must answer the following two important questions: How to establish effective communication mechanisms to get such users involved? How to effectively transfer usability knowledge to contributing users?

A limitation of our experiment is that only one open source project was studied, which clearly may not reflect the comprehensive situation of the entire open source community. Therefore, future studies will not only expand the sample size, but also the range of open source projects from various categories.

APPENDIX 1: USABILITY PROBLEMS FOUND BY THE CASE STUDY

A1. There should be redo and undo functionality that stores the last x number of actions so that users can go back several steps when they realize their mistake or go forward if they want to.

A2. There should be an ability to drag and drop tasks.

A3. When a new task is created, a pop up box should ask the user at what level to create the task. Currently by default it is created as a sub task of the currently selected task. This could get confusing.

A4. The time scrolling function by dragging the mouse in the right pane can be handy, but can also get confusing and frustrating.

A5. Because of cut being the first item on the context menu both my subjects inadvertently left clicked to close the context menu and deleted the task.

A6. Open command always defaults to MyDocuments even when you have already opened or saved to another directory. It should default to the most recent directory at least within a session.

A7. The interface for selecting resources is somewhat confusing. It is hard to realize that there is a drop-down box for selecting existing resources. People tend to type in directly the resource name in the ID box.

A8. The Duration textbox should prevent users from entering non-numerical information.

A9.It is a little redundant and confusing to have "start-date", "end-date" and "duration" fields for each task and it seems to be a bug that all other fields do not change correspondingly and correctly when any one of them is changed.

A10. To modify an existing resource, users should be able to open the Property dialog box by double-clicking. (Nichols and Twidale 2003)

REFERENCES

1. Boren, T., and Ramey, J. (2000). Thinking aloud: reconciling theory and practice. *IEEE Transactions on Professional Communication*, 43, 3, 261-278.
2. Hilbert, D. M. and D. F. Redmiles (2000). "Extracting usability information from user interface events." *ACM Computing Surveys (CSUR)*, 32, 4, 384-421.
3. Jeffries, R., J. R. Miller, et al. (1991). User interface evaluation in the real world: a comparison of four techniques. *SIGCHI conference on Human factors in computing systems: Reaching through technology*.
4. McCoy, T. (2002). "The Politics of Usability." *The Usability SIG Newsletter*, 8, 4.
5. Messerschmitt, D. G. (2004). "Back to the user [open source]." *IEEE Software*, 21, 1, 89-91.
6. Nichols, D. M., D. McKay, et al. (2003). Participatory Usability: supporting proactive users. *opensource.mit.edu*.
7. Nichols, D. M., K. Thomson, et al. (2002). Usability and open-source software development. *opensource.mit.edu*.
8. Nichols, D. M. and M. B. Twidale (2003). "The Usability of Open Source Software." *First Monday*, 8, 1.
9. Pemberton, S. (2004). "Scratching Someone Else's Itch (Why Open Source Can't Do Usability)." *Interactions*, 11, 1, 72.
10. Salmoni, A. J. (2004). Open source usability, *http://www.advogato.org/article/780.html*.
11. Shneiderman, B. (1998). Designing the user interface: strategies for effective human-computer-interaction, Addison Wesley Longman, Inc.
12. Someren, M. v., Barnard, Y., and Sandberg, J. (1994). The Think Aloud Method: A Practical Guide to Modelling Cognitive Processes: London: Academic Press.
13. Twidale, M. B. and D. M. Nichols (2004). Usability Discussions in Open Source Development, University of Waikato, Hamilton, New Zealand.
14. van Welie, M. and G. C. van der Veer (2003). Pattern Languages in Interaction Design: Structure and Organization. *Ninth IFIP TC13 International Conference on Human-Computer Interaction*.
15. Zhao, L. and S. Elbaum (2003). "Software Quality Assurance under the Open Source Model." *Journal of Systems and Software*, 66, 1, 65-75.