Communications of the Association for Information Systems

Volume 20 Article 25

October 2007

Waiting for Usable Open Source Software? Don't Hold Your Breath!

Ravi Sen
Texas A&M University, rsen@mays.tamu.edu

Communications of the Association for Information Systems

WAITING FOR USABLE OPEN SOURCE SOFTWARE? DON'T HOLD YOUR BREATH!

Ravi Sen
Information and Operations Management Department
Mays Business School
Texas A&M University
rsen@mays.tamu.edu

ABSTRACT

There is a general consensus about the lack of usability in most open source software (OSS). Academics and practitioners have offered several suggestions to improve the usability of such software. However, a realistic assessment of OSS projects, specifically the motivations of OSS developers and their attitude toward software usability, lack of user feedback, and absence of usability experts in OSS projects, leads to the conclusion that strategies to improve OSS usability are unlikely to succeed anytime soon. The only exceptions will be OSS which enjoy sufficient financial support from individuals and organizations, and software that were developed by commercial software producers and later released under an open source license.

.**Keywords**: software usability, OSS, FLOSS, usability

I. INTRODUCTION

An open source software (OSS) allows users to have access to the source code of the software, the freedom to use the software as they see fit, improve it, fix its bugs, augment its functionality, and redistribute the software under an OSI[1] approved license to other users for free or at a charge, who could themselves modify and/or use it according to their own needs.[2] For those who are interested in knowing more about OSS, AlMarzouq et al. [2005] offer an excellent tutorial on the subject. The tutorial provides detailed discussions on OSS development, OSS licensing, and benefits of OSS. In addition, there is a huge body of research on OSS, which focuses on motivations of OSS developers, success of OSS, and the impact of OSS on software markets [see Niederman et al*.* 2006a and 2006b; Sen 2007; Nelson et al*.* 2006 for OSS research classification]. Interestingly, the success of OSS has been mostly confined to technically skilled or "power" users, who use OSS in their own software development projects or as part of the larger computing infrastructure. For example, Apache, Linux, and Sendmail, considered successful OSS, are generally adopted by users with high software skills. An overwhelming majority among average computer users, who have limited software skills, still rely on proprietary and commercial software [Lerner and Tirole 2002]. One of the main reasons cited for the limited acceptance of open source software among the average users is the lack of usability in OSS. For instance, in a

[1] Open Source Initiative

[2] http://opensource2.planetjava.org/docs/definition.php

survey conducted by EU on open source software, usability was not seen as an advantage of open source software over proprietary software.[3] For example, desktop versions of UNIX, an open source operating system, have relatively poor usability in comparison to closed-source commercial operating systems such as Windows XP. This poor usability of UNIX is best illustrated by the following excerpt from an essay by one of the strongest proponents of OSS-

> I've just gone through the experience of trying to configure CUPS, the Common UNIX Printing System. It has proved a textbook lesson in why non-technical people run screaming from UNIX. This is all the more frustrating because the developers of CUPS have obviously tried hard to produce an accessible system - but the best intentions and effort have led to a system which despite its superficial pseudo-friendliness is so undiscoverable that it might as well have been written in ancient Sanskrit. [Eric Raymond 2004].

In response to this essay, Raymond received several letters from the open source community. Many of the writers, who considered themselves to be skilled software users, had found themselves in similar situations when using open source software. The problem, therefore, is not just that an average non-technical user does not find open source software usable - the problem is that even “expert users” such as Eric Raymond sometimes have trouble with OSS usability. In short, while open source software has gained a reputation for reliability, efficiency, and functionality, its poor usability forms a major obstacle in its widespread use. The significance of OSS usability (or lack of it) can be gauged from the numerous studies and commentaries addressing the issue [e.g. Behlendorf 1999; Raymond 2004; Manes 2002; Nichols et al. 2001; Thomas 2002; Frishberg et al*.* 2002]. These studies offer several suggestions and strategies that could lead to improvements in OSS usability. In this paper we analyze these suggestions and strategies to determine if they can be successfully implemented in OSS development. The paper is structured as follows. Section III summarizes the popular usability testing method, which are then analyzed to determine their applicability for open source software. Section IV provides an analysis of usability improvement strategies that have been proposed in the current literature, followed by the conclusion in Section V.

II. SOFTWARE USABILITY TESTING METHODS

Before any improvements can be made to OSS usability, the OSS developers need to first test their software for usability. Current usability testing techniques are generally classified into two basic types: inspection and empirical [Holzinger 2005]. Inspection methods are conducted by usability specialists such as HCI[4] and the software design team, and are useful for identifying major flaws in an interface before the software reaches the end users. Empirical tests can be conducted with actual members of the target user population [Nielsen 1993]. It involves understanding the potential non-developer users’ characteristics such as their computer skills, work experience, educational levels, and ages. An understanding of these characteristics helps the software developers to anticipate the users’ ability to learn and use the software, and then modify the software complexity to make it usable for these potential users. Table 1 provides a brief description of popular usability inspection methods that are currently in use.

CAN EXISTING USABILITY TESTING METHODS WORK FOR OSS?

Before we can address this question we need to determine the inherent requirements for implementing the existing usability testing methods (Table 1). As we can see from Table 1, the empirical methods for testing usability involve usability experts and end-users. Among inspection

[3] http://flosspols.org/deliverables/D03HTML/FLOSSPOLS-D03 local governments survey reportFINAL.html

[4] Human Computer Interface

methods, some involve a representative sample of potential users (i.e. Pluralistic Walkthrough and Feature Inspections), while most involve HCI experts (i.e. Heuristic Evaluation; Guideline Review; Pluralistic Walkthrough; Standards Inspections; Cognitive Walkthroughs; Formal

Table 1. Summary of Software Usability Testing Methods [Holzinger 2005]

Empirical Methods		
Method Name	**Brief Description**	**Who is Involved?**
Empirical Usability Testing	Understanding the potential non-developer users' characteristics that can help the developers to anticipate the users' ability to learn and use the software.	Usability Experts and end-users
Inspection Methods		
Method Name	**Brief Description**	**Who Is Involved?**
Heuristic Evaluation	An informal way to determine whether the interface conforms to established usability principles	Usability specialists, e.g. HCI experts
Guideline Review	A complex method in which the interface is tested for conformance with a comprehensive list of usability guidelines	Usability specialists, e.g. HCI experts
Pluralistic Walkthrough	This method involves following a scenario [e.g. a possible software use], and the discussion of potential usability issues.	Representative users; Developers; and HCI experts
Consistency Inspections	Software developers meet to see whether an interface's behavior is consistent with their designs.	OSS Developers
Standards Inspections	An expert inspects an interface for compliance with industry standards. These evaluations are designed to increase the usability of an interface in comparison with other systems on the market that follow the same set of standards	Usability specialists, e.g. HCI experts
Cognitive Walkthroughs	Simulate users' problem-solving processes. This test evaluates whether the simulated user's goals lead from one action to the next correctly.	Usability specialists, e.g. HCI experts
Formal Usability Inspections	A moderator is appointed to manage inspections and the inspection meeting. A design owner [e.g. project leader] is responsible for design and redesigns. Inspectors find problems with the interface; and a scribe records all issues identified during the meeting. Formal inspections use the following process: planning, a preliminary meeting, a preparation phase where inspectors review the interface, an inspection review where lists of usability problems are merged, and a follow-up phase where the effectiveness of the inspection process itself is assessed.	Usability specialists, e.g. HCI experts; and OSS Developers
Feature Inspections	These evaluations test the functionality delivered in software, and assess whether it meets the needs of the users.	Representative users; and OSS Developers

Usability Inspection). Some methods require the participation of both HCI experts and a representative sample of potential software users. Therefore, it is logical to conclude that in order for these usability testing methods to be applicable in the context of open source software, OSS project administrators need to involve either users, or usability experts, or both in their projects. However, at present most OSS projects lack both active non-developer user participation and usability-experts' participation.

Generally speaking most non-developer users[5] are passive members of the open source community. For example about 99 percent of people who use Apache are passive users [Nakakoji 2002]. They make no contribution to the OSS code, and do not volunteer to support the project in any way or manner. Even when they are dissatisfied by the software, they are more likely to use the next available alternative rather than solve the problems that they have with the software. In such circumstances, it would be unrealistic to expect user involvement in usability testing exercise. Therefore, the usability testing methods involving users do not seem to be suitable for OSS. Usability experts do not get involved in OSS projects for two reasons. First, most OSS developers believe that usability is purely a technical issue and as such can be taken care of by the developers themselves without any help from usability experts [Englich 2004]. Therefore, they are reluctant to involve usability experts in their OSS development projects. Second, even when OSS developers are not against seeking the help of usability experts, they do little to welcome these experts to their OSS projects, as is evident from the following quote by a usability expert:

> Raymond and his ilk have no respect for anyone but themselves. They have no respect for the fact that UI design is a special talent. They have no respect for the fact the good UI design requires a tremendous amount of time and effort. And, most importantly, they have no respect at all for real users. [Ronco Spray 2004]

Third, the incentives in the OSS community are geared toward increasing developer participation, i.e. they work better for improvement of functionality than of usability [e.g. Feller and Fitzgerald 2002; Hars and Ou 2001]. Since the usability experts do not add any feature to the OSS, their contribution is ignored by the OSS developer community. Therefore, these experts have minimal incentive to participate in open source projects [e.g. Nichols and Twidale 2003]. Therefore, it would be unrealistic to expect help from usability experts in testing OSS usability.

In summary, in order to test OSS for usability, the OSS developers need active participation from non-developer users and usability experts (i.e. HCI experts). However, for reasons discussed earlier, both are most likely to abstain from active participation in OSS usability testing. Therefore, the key challenge before OSS projects is to ensure active participation from non-developer users and HCI experts before they can move on to the next step of actually improving the usability of their software. Nichols and Twidale [2003] have suggested several strategies to involve users and usability experts in OSS usability testing. Furthermore, initiatives to match OSS developers with usability experts have been also been undertaken (e.g. see openusability.org). In the following section we will analyze these strategies and initiatives to determine if would be successful.

III ANALYSIS OF STRATEGIES FOR INVOLVING USERS AND USABILITY EXPERTS IN OSS PROJECTS

All software usability testing techniques involve the users of the software, usability experts (e.g. HCI experts), or both. The current literature suggests that to encourage user-participation in usability testing, OSS administrators need to provide tools that reduce the effort required on the part of users to offer their suggestions and feedback on usability issues, and provide financial incentives to the users who provide the feedback or participate in usability testing studies. Usability experts can be encouraged to participate in OSS usability testing by offering them due

[5] Non-developers Users- Users who use OSS but do not contribute any code to the OSS.

recognition and awarding their contributions to OSS projects. This section analyses these strategies to determine the likelihood of their success.

GETTING NON-DEVELOPER USERS INVOLVED

Red Hat's Havoc Pennington and Novell's Jimmac suggest[6] the following:

> …users write an analysis and test cases of a feature request the user wants to see implemented, because this way they might get the developer motivated to actually implement it.

This is not an easy advice to implement, since even a small amount of extra effort is enough to discourage users from providing any feedback or suggestion [Nielsen 1993]. Therefore, expecting users to write detailed test cases is unrealistic. Furthermore, there are two problems with this approach [Eugenia Loli-Queru 2005]:

1. Most non-developer users are not skilled enough to write a test case.
2. Average users do not use bug reporting tools such as bugzilla,[7] mainly because they do not like spending time to register in order to be able to use it.

Therefore, to encourage feedback from non-developer users, OSS developers need to ensure that *these users have to apply minimal effort to participate in usability testing* [Nielsen 1993]. Nichols and Twidale [2003] draw on the existing work in the field of HCI (Human Computer Interface) and offer several excellent suggestions to achieve this. For example, an OSS can have features or tools that allow users to report any usability problems while they are using the OSS. Existing HCI research [Hartson and Castillo 1998; Thompson and Williges 2000] has shown, on a small scale, that user reporting is effective at identifying usability problems. In addition to these user-initiated reports, applications can prompt users to offer feedback on the basis of their experience with software [Ivory and Hearst 2001]. These feedbacks from individual users can be combined and interpreted to provide critical usability information that can be used to improve the software [Nielsen 1993]. OSS developers can also use the knowledge accumulated in the area of remote usability [Hartson et al*.,*1996; Scholtz, 2001], to distribute the burden of providing usability reports among several users. However, the OSS developers will need to ensure that they are able to coordinate these studies and interpret the results. While all these suggestions are noteworthy, I believe that there are challenges to implementing them in OSS projects.

Before these suggestions can be implemented, it would require the addition of new features to the OSS [e.g. to coordinate the distributed usability reports; and to prompt the user to provide feedback about the software's usability] resulting in more software coding. However, OSS developers generally avoid "code bloating." These developers accept new features only if the features are deemed absolutely necessary for the OSS. In fact "lean-and-mean" code is considered one of the advantages of OSS over the closed-source commercial software.[8] Furthermore, even if these additional features, that allow users to provide feedback on usability, are implemented, there is no guarantee that software usability will improve because the developers might not act on the information/ feedback/comment received from the users. This inaction could be because of resource (e.g. time, developers) constraints, or because the OSS developers are not interested in acting on them. There is some evidence to suggest that OSS projects often ignore requests for new features from *non-developer users* [Eugenia Loli-Queru 2005]. For example, the key Gnome feature request, a usable menu editor, has yet to be offered after so many years. In some cases (e.g. Red Hat Linux), the developers care only if the

[6] http://www.osnews.com/story.php?news_id=9933

[7] http://www.bugzilla.org/

[8] http://www.tbs-sct.gc.ca/fap-paf/oss-ll/foss-llo/foss-llo10_e.asp

feedback comes from their marketing departments. This apathy toward the needs of the non-developer user is best illustrated by the following quotations:

> A feature will be implemented if and only if there is a developer who wants to implement it. A Gnome developer [see Eugenia Loli-Queru, 2005].
>
> KDE will be able to sustain itself just fine without users, while it will not last a single day without developers. So when it comes to choosing between scaring away developers and scaring away users, the choice is rather easy actually.. Waldo Bastian, SuSE Linux [one of the replies to Eugenia Loli-Queru's article [2005][9]

Finally, there is the issue of incentive for users to provide feedback on OSS usability. Some suggest that users who provide the feedback could be financially rewarded for such contributions, such as discounts on future software purchases [e.g. Shneiderman 2002]. While this strategy might work for commercial software projects, it will not work as well in case of OSS projects because most of these projects (except those supported/sponsored by commercial organizations) are already resource constrained, and therefore do not have the finances required to reward users. In short, *the suggestions to involve users in usability testing would be extremely difficult to implement.* What about involving usability experts?

INVOLVING USABILITY EXPERTS

Many of the usability inspection techniques can be implemented by involving usability (i.e. HCI) experts. One way to do this is to partner with commercial software producers [Nichols and Twidale 2003], and leverage their expertise in developing usable software. Furthermore, these commercial partners can provide the resources necessary to carry out empirical usability studies. However, there can be conflicts of interest and misunderstandings between the commercial partners and the OSS developers about the direction of interface development [Trudelle 2002]. For example, if the commercial partner is interested in selling a bundle of OSS plus support-service (e.g. Red Hat Linux), it is better off by ensuring that the freely available OSS remains less usable in comparison to the commercial version of the same OSS [Sen 2006]. Therefore, OSS developers involved with such projects have little incentive to seek help from commercial partners to hire usability experts.

Another approach to get usability experts involved with OSS projects is to make the OSS projects attractive to them so that they volunteer their services for free. However, for this to happen, these usability experts will need to feel welcomed and valued. OSS developers will have to recognize the fact that software usability is more than just a pretty user interface [e.g. Benjamin 2005]. They will have to accept it as an important characteristic of software, and respect professionals who specialize in the area, even if these experts lack the software development skills of the OSS developers. References to *clueless newbies* and *lusers,* and some of the more offensive language used to describe usability experts will need to be curtailed. In addition, these experts can be made to feel welcome by [Nichols and Twidale 2003]: (a) recognizing the importance of usability issues within the OSS community; (b) providing them with tools that they can use to offer their input on usability,[10] e.g. enabling them to be able to talk productively to each other [Crabtree et al. 2000]; (c) putting in place a well-defined procedure to handle any problems arising from conflicts between proposed usability improvements OSS functionality; and (d) recording the contributions of usability experts in the evolution of the software. While these incentives might attract usability experts to OSS projects, the success of this initiative in improving OSS usability *depends upon the actions taken by the OSS developers in response to the suggestions offered*

[9] http://osnews.com/story.php?news_id=9953

[10] Hartson and Castillo [1998] review various graphical approaches to bug reporting including video and screenshots, which can supplement the predominant text-based methods.

by these usability experts. At present the evidence suggests that the OSS developers are unlikely to act on the suggestions received from the usability experts. For example, there are about 100,000 OSS projects registered at Sourceforge. However, as of 17 April 2007, *only 200 projects are registered at openusability.org, a site for projects looking for usability advice and interaction designers who want to help*. One interpretation of this statistics is that most OSS projects are not yet interested in getting advice on usability. Therefore, one can safely conclude that that it would be difficult to implement any usability inspection method that requires the involvement of usability experts.

Given our conclusion that involving end-users and usability experts in usability testing is easier said than done, what other options do we have to improve OSS usability? Nichols and Twidale [2003] suggest the use of automated usability evaluation (AUE) techniques [e.g. Ivory and Hearst 2001], or use of "expectation agents" [Hilbert and Redmiles 2001] to understand potential user behavior. Understanding of potential user behavior can help the OSS developers improve software usability. However, this approach assumes: (a) that OSS developers understand the significance of usability for non-developer users, (b) they are skilled in the use of these automatic tools to evaluate usability, and they know the proper use if these tools. Let us address these assumptions one by one.

OSS DEVELOPERS AND USABILITY

OSS developers are known to emphasize function over form. Most OSS projects start because there is a need for some software and this need is not fulfilled by the existing software. As a result, they go ahead and develop the software that meets these needs [Shah 2006]. Therefore, the immediate priority of OSS developers is to get the functionality right. Thus, most OSS projects are driven by functional requirements of developers.[11] One could, in fact, argue that OSS is a prime example of user-centric design (UCD): a framework which recognizes that a good understanding of the needs of core users is integral to producing usable software [Benjamin 2001]. In case of OSS, since the developers are also the core users of the software [Mockus et al. 2002], they tend to look at usability from their own perspective. For these *developer-users* the OSS application is already usable because they know the software inside out, and can always modify it as they choose fit. Given this narrow perspective about usability on the part of OSS developers, it is unrealistic to assume that most OSS developers will understand the significance of usability for *non-developer users*.

OSS DEVELOPERS AND AUTOMATED USABILITY EVALUATION (AUE) TECHNIQUES

As the name suggests, automated usability evaluation tools automate various aspects of usability evaluation such as capture and measurement of usability features, their analysis, to identification of usability problems, and suggestions to improve usability [Balbo 1995]. For a better understanding of current AUE tools please refer to the survey of 128 AUE methods by Ivory and Hearst [2001]. Given the fact that most OSS projects involve only a few developers [Krishnamurthy 2002], who come together to develop a small piece of software that performs a specific task, and then move on, it is unrealistic to expect these developers to spend time learning about automated usability evaluation tools and techniques. Furthermore, most OSS projects might not have the financial resources to use such tools. Finally, the use of these tools and techniques will require OSS developers to act upon the feedback provided by these tools. As we have argued earlier, OSS developers might be reluctant to do so because (a) they do not want to add features that are not absolutely necessary, i.e., avoid code bloating; and (b) they do not have the time to address issues that concern only non-developer users, which is often the case since most OSS developers work on OSS projects as volunteers in their spare time.

[11] We will refer to these individuals as either developers or developer-users in the remainder of the paper. All other users will be referred to simply as users or non-developer users.

IV. CONCLUSION

Most of the suggestions offered for improving the usability of OSS are based on the assumption that as long as they are “compatible” with the distributed methodology of developing OSS they should be applicable to OSS development. However, a qualitative analysis of the behavioral aspects of OSS development, more specifically those of OSS developers leads us to conclude otherwise. Therefore, educating OSS developers about the importance of usability might be the most important thing that needs to be accomplished [Nichols and Twidale 2003] before we can expect any improvement in OSS usability. However, this would not be an easy task. Proponents of better usability have tried to convince open source developers that a usability approach to their development would be a good thing, but with little success. The responses are typically one of the following: [12]

- I've been a developer for *x* years and I think I know what I'm doing.
- I know what the users want.
- I'm doing this for myself [a valid response].
- If I ignore you, will you go away?
- Submit a coding change and we'll look at it.

Since usability is very important for non-developer users, OSS developer need to be convinced that usability will increase the user-base of the software. With this strategy, at least those OSS developers who are motivated by the size of the potential installation base of their OSS [e.g. Nickell 2001] would be interested in improving the usability of their software. However, most OSS developers might not be interested in a large installed base of their OSS. For instance, they may consider their project a success if the OSS meets their unique functional requirements, if it is used by other OSS projects, or if it attracts a large number of developers [e.g. Stuart et al. 2006; Crowston et al. 2003]. These OSS developers are less likely to worry about the usability concerns of non-developer users. In some cases where the OSS developers accept the significance of usability they believe that they can take care of usability concerns without involving non-developer users or usability experts. While an exceptional software developer can both design and code usable software, the majority do not have the necessary skills to undertake such a project. Most developers lack an understanding of interface types and interface design models, and the application of these models [Smith 2002]. Finally, most OSS developers work under a license that prevents them from selling the code, so they develop features that are important to them and not necessarily to the market.

In short, though theoretically possible, a realistic assessment of OSS projects shows that strategies to improve OSS usability are unlikely to succeed soon because usability expert have little incentive to get involved with OSS projects; non-developer users find it easier to move on to a more usable software instead of providing meaningful feedback to OSS developers; and most OSS developers are yet to be convinced about the significance of usability for non-developer users. The only exceptions will be (a) those OSS that enjoy sufficient financial support from individuals and organizations to invest in the extra resources needed to carry out usability testing and then implementing the results obtained from these testing (e.g. Red Hat Linux, SuSE Linux); and (b) software that was developed by commercial software producers and later released under an open source license (e.g. Firefox). Therefore, individuals and organizations who value usability in their software should either opt for commercial versions of open source software or purchase closed-source software from commercial software producers.

12 http://www.usabilityprofessionals.org/upa_publications/upa_voice/volumes/5/issue_1/open_source.htm

REFERENCES

AlMarzouq, M., L. Zheng, G. Rong, and V. Grover. (2005). "Open Source: Concepts, Benefits, and Challenges," *Communications of AIS*, (16)37, November, pp. 1-49.

Balbo, S. (1995). "Automatic Evaluation of User Interface Usability: Dream or Reality," In S. Balbo, Ed., Proceedings of the Queensland Computer- Human Interaction Symposium (Queensland, Australia, August). Bond University.

Behlendorf, B. (1999). "Open Source as a Business Strategy," In: M. Stone, S. Ockman, and C. DiBona (eds.) *Open Sources: Voices from the Open Source Revolution.* Sebastopol, California: O'Reilly & Associates, pp. 149-170.

Benjamin, D. (2001). "Designing for Usability," *TEST FOCUS*, (2)2, http://www.testfocus.co.za/Feature%20articles/Feb2001.htm (current February 27th 2007).

Benjamin, R. (2005). "Usable GUI Design: A Quick Guide for F/OSS Developers," http://benroe.com/files/gui.html (current February 27 2007).

Crabtree, A., D. M. Nichols, J. O'Brien, M. Rouncefield, and M. B. Twidale. (2000). "Ethnomethodologically Informed Ethnography and Information System Design," *Journal of the American Society for Information Science*, (51) 7, pp. 666-682.

Crowston, K., H. Annabi, and J. Howison. (2003). "Defining Open Source Software Project Success," In Proceedings of the 24th International Conference on Information Systems (ICIS 2003), Seattle, WA.

Englich, F. (2004). "Open Source Usability Is a Technical Problem We Can Solve on Our Own," http://programming.newsforge.com/programming/
04/07/07/1640244.shtml (current Friday July 09, 2004).

Eugenia Loli-Queru. (2005). "OSS Software, Deaf Developers & Unsatisfied Users," http://www.osnews.com/story.php?news_id=9933 (current 10 March 2003).

Feller, J. and B. Fitzgerald. (2002). *Understanding Open Source Software Development*. London: Addison-Wesley.

Frishberg, N., A. M. Dirks, C. Benson, S. Nickell, and S. Smith. (2002). "Getting to Know You: Open Source Development Meets Usability," *Extended Abstracts of the Conference on Human Factors in Computer Systems (CHI 2002).* New York: ACM Press, pp. 932-933.

Hars, A. and S. Ou. (2001). "Working for Free? — Motivations of Participating in Open Source Projects," Proceedings of the 34th Annual Hawaii International Conference on System Sciences. New York: IEEE Computer Society Press, pp. 2284-2292.

Hartson, H. R. and J. C. Castillo. (1998). "Remote Evaluation for Post-Deployment Usability Improvement," Proceedings of the Conference on Advanced Visual Interfaces (AVI'98), New York: ACM Press, pp. 22-29.

Hartson, H. R., J. C. Castillo, J. Kelso, W. C. Neale. (1996). "Remote Evaluation: The Network as an Extension of the Usability Laboratory," Proceedings of the Conference on Human Factors in Computing Systems (CHI'96), New York: ACM Press, pp. 228-235.

Hilbert, D. M. and D. F. Redmiles. (2001). "Large-Scale Collection of Usage Data to Inform Design," In: M. Hirose (ed.). Human-Computer Interaction — INTERACT'01: Proceedings of the Eighth IFIP Conference on Human-Computer Interaction, Tokyo, Japan, pp. 569-576.

Holzinger, A. (2005). "Usability Engineering Methods for Software Developers," *Communications of the ACM*, (48)1, pp 71-74.

Ivory, M. Y. and M. A. Hearst. (2001). "The State of the Art in Automated Usability Evaluation of User Interfaces," *ACM Computing Surveys*, (33) 4, pp. 470-516.

Krishnamurthy, S. (2002). “Cave or Community? An Empirical Examination of 100 Mature Open Source Projects,” First Monday, (7) 6, http://firstmonday.org/issues/issue7_6/krishnamurthy/ (current 27 February 2007).

Lerner, J. and J. Tirole (2002). “Some Simple Economics of Open Source,” *Journal of Industrial Economics*, (46)2, 125-156.

Manes, S. (2002). "Linux Gets Friendlier," *Forbes* (10 June), pp. 134-136.

Mockus, A., R. Fielding, and J. Herbsleb. (2002). “A Case Study of Open Source Software: The Apache Server,” In *Proceedings of 22nd International Conference on Software Engineering*, Limmerick, IR, pp 263-272.

Nakakoji, K., Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye. (2002). "Evolution Patterns of Open-Source Software Systems and Communities," Proceedings of the Workshop on Principles of Software Evolution, International Conference on Software Engineering. New York: ACM Press, pp. 76-85.

Nelson, M., R. Sen, and C. Subramaniam. (2006). “Understanding Open Source Software Development: A Research Classification Framework,” *Communications of AIS*, (17)12, (February), pp. 266-287.

Nichols, D. M., K. Thomson, and S. A. Yeates. (2001). "Usability and Open Source Software Development," In E. Kemp, C. Phillips, Kinshuck, and J. Haynes (eds.). Proceedings of the Symposium on Computer Human Interaction. Palmerston North, New Zealand: SIGCHI New Zealand, pp. 49-54.

Nichols, D. M. and M. B. Twidale. (2003). “The Usability of Open Source Software,” *First Monday* 8(1). http://www.firstmonday.org/ issues/issue8_1/nichols/ (current 27 February 2007).

Nickell, S. (2001). "Why GNOME Hackers Should Care about Usability," GNOME Usability Project, http://developer.gnome.org/ projects/gup/articles/why_care/, (current 27 February 2007).

Niederman, F., A. Davis, M. E. Greiner, D. Wynn, and P. T. York. (2006a). “A Research Agenda for Studying Open Source I: A Multi-level Framework,” *Communications of AIS*, (18)7, August, pp.1-38.

Niederman, F., A. Davis, M. E. Greiner, D. Wynn, and P. T. York. (2006b) “A Research Agenda for Studying Open Source II: View through the Lens of Referent Discipline Theories,” *Communications of AIS*, 18, August, pp.150-175.

Nielsen, J. (1993). *Usability Engineering*, San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Raymond E. (2004). “The Luxury of Ignorance: An Open-Source Horror Story,” http://people.lis.uiuc.edu/~twidale/research/ossui/ (current 03 Jul 2004).

Raymond, E. S. (1999). "The Revenge of the Hackers," In M. Stone, S. Ockman, and C. DiBona (eds.) *Open Sources: Voices from the Open Source Revolution.* Sebastopol, Calif.: O'Reilly & Associates, pp. 207-219.

Scholtz, J. (2001). "Adaptation of Traditional Usability Testing Methods for Remote Testing," Proceedings of the 34th Annual Hawaii International Conference on System Sciences. New York: IEEE Computer Society Press.

Sen, R. (2006). “A Strategic Analysis of Competition between Open Source and Proprietary Software,” Forthcoming in *Journal of Management Information Systems*, 2007.

Shah, S. (2006). “Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development,” *Management Science*, (52) 7, pp 1000-1014.

Shneiderman, B. (2002). *Leonardo's Laptop: Human Needs and New Computing Technologies*, Cambridge, Mass: MIT Press.

Smith, P. (2002). “Debunking the Myths of UI Design,” http://www-128.ibm.com/developerworks/library/us-myth.html (current 27 February 2007).

Spray, R. (2004). “Ronco Spray on Usability,” http://daringfireball.net/2004/04/spray_on_usability (current 27 February 2007).

Stewart, K. J., A. P. Ammeter, and L. M. Maruping. (2006). “Impact of License Choice and Organizational Sponsorship on Success in Open Source Software Development Projects,” Forthcoming in *Information System Research*.

Thomas, M. (2002). "Why Free Software Usability Tends to Suck," http://web.archive.org/web/20041117091141/http://mpt.phrasewise.com/discuss/msgReader$173 (current February 2007).

Thompson, J. A. and R. C. Williges. (2000). "Web-Based Collection of Critical Incidents during Remote Usability Evaluation," Proceedings of the 14th Triennial Congress of the International Ergonomics Association and the 44th Annual Meeting of the Human Factors and Ergonomics Society (IEA 2000/HFES 2000), Santa Monica: Human Factors and Ergonomics Society, Volume 6, pp. 602-605.

Trudelle, P. (2002). "Shall We Dance? Ten Lessons Learned from Netscape's Flirtation with Open Source UI Development," presented at the Open Source Meets Usability Workshop, Conference on Human Factors in Computer Systems (CHI 2002), Minneapolis, Minn., http://www.iol.ie/~calum/chi2002/peter_trudelle.txt (current 27 February 2007)

ABOUT THE AUTHOR

Ravi Sen is an assistant professor in the Department of Information and Operations Management at Mays Business School, Texas A&M University. He received his Ph.D in Business Administration (Major: MIS) in 2003 from the College of Business, University of Illinois at Urbana-Champaign. His research interests include open source software, electronic commerce, and software security. He has published in *Journal of Management Information Systems (JMIS), International Journal of Electronic Commerce (IJEC)*, *Communications of AIS (CAIS)*, and *Electronic Markets*.

Communications of the Association for Information Systems

ISSN: 1529-3181