Assuring Quality and Usability in Open Source Software Development

Henrik Hedberg, Netta Iivari, Mikko Rajanen & Lasse Harjumaa
Department of Information Processing Science, University of Oulu
{henrik.hedberg, netta.iivari, mikko.rajanen, lasse.harjumaa}@oulu.fi

Abstract

This paper reviews literature on quality and usability assurance in open source software (OSS) development, focusing specifically on OSS that is targeted at a large user population, which does not consist only of OSS developers anymore. In this type of OSS development, the 'naïve', non computer professional users should be taken into account and usability of OSS improved. Furthermore, software quality becomes a very relevant issue to be assured. We contrast OSS literature on quality and usability with prescriptive literatures derived from the fields of software engineering and human computer interaction. We present a summary of the current practices utilized to assure quality and usability in OSS development, and recommend practices to be used in this context. We also point out limitations in the existing research and suggest paths for future work.

1. Introduction

In this paper we review literature on quality and usability assurance in open source software (OSS) development, focusing specifically on OSS that is targeted at a large user population, which does not consist only of OSS developers themselves anymore. In all, the user base of OSS is becoming larger and larger, including a growing number of non-technical users. In addition, during recent years firms have started to consider how to gain competitive advantage from OSS [8]. Software (SW) companies are releasing the source code of their commercial products and participating in OSS communities developing it further [8, 16]. OSS is, therefore, no longer developed only to serve particular developers and their needs [8]. Instead, there will be more and more users without deep technical knowledge. That leads to higher quality expectations than earlier, when an end-user was seen as a co-developer tolerating crashes, hunting defects, and fixing the code.

In this paper it is acknowledged that the end user population - including the 'naïve', non computer professional users - is constantly growing, and therefore usability of the OSS should be improved [2, 7, 17, 19, 22, 23]. SW quality, altogether, becomes a very relevant issue to be assured. Typically OSS developers rely on the large user base reporting defects and the existence of volunteering co-developers that can be compared with beta-testing. However, in this new situation, quality assurance must be done before the SW is delivered to the end-users, who may also pay for it and thus expect smooth operation.

Therefore, we review OSS literature on quality and usability assurance and contrast it with prescriptive usability and quality assurance literatures derived from the fields of software engineering (SE) and human computer interaction (HCI), within which this kind of issues have been considered as legitimate concerns for decades. As a result, we present a summary of the practices currently utilized to assure quality and usability in OSS development. In addition, we propose practices that could be used to produce high quality and more usable OSS - especially for the non technical users. Finally, we will also point out that the existing research is somewhat limited, and thus suggest new, interesting research topics to be explored.

2. Quality Assurance in OSS Development

The purpose of quality assurance in SW development is to ensure the high quality of both the final product and the process used to produce those products. OSS development seems to challenge traditional ways to achieve that.

2.1 Current research and practice

There are many evidences that OSS development can produce high quality SW, such as Linux, and Apache. However, the outcome of the OSS development process depends on the skills of the participating developers. There is no strict process to follow, design and planning are rare, development process is ad hoc style, and quality assurance

techniques are mainly ignored, except for beta-testing with the extension of the ability to review and fix the actual source code [24]. The projects have been successful, because developers were using the SW themselves, which motivates them and gives them excellent domain area knowledge [20]. It has been argued that as volunteering professionals the OSS developers belong to the top five percent of all SW developers what comes to talent [9].

Typically, there is a lead developer or a small amount of developers forming a core team that controls the overall architectural design and the course of an OSS project [7, 15]. Other developers participate by reading the code and delivering patches, i.e. modified source code extracts, to the core team [20]. It is obvious that managing volunteering co-developers without face to face contact by means of, for example, discussion forums and email, necessitates good social and communication skills [20].

The core team reviews patches and decides if those are accepted. Typically the majority of modifications requests delivered by users are not integrated into the product. The Apache project is an extreme example of this, because every patch is reviewed in developers' mailing list by at least the core team. [15] In general, in OSS development it is rarely required that the all of the code is reviewed, but instead one relies on extensive real-world testing by users and co-developers [24].

Users' role is to find and report defects, which are then fixed by developers not necessarily belonging to the core team. Fast-paced release cycle satisfies users and ensures that debugging efforts are not thrown away. [20] However, most of the users are simply using the application and not reviewing the code before they mobilize it [9]. Therefore, users are plain beta testers, who can only report annoying defects, if they have enough interest and feel that the application is worth it. Rarely users are skilled enough to locate the defects and even less of them are able to fix them. According to Linus Law, "given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone." [20] At the same time, this implies the existence of the problem in the application already employed by a large number of users.

Testing is very limited in OSS projects. Usually there is no testing plan, and the test coverage is not addressed. In addition, despite OSS development seems to be quite tool-centered, a relative small number of projects utilize testing tools. However, testing takes a significant portion of SW life time. [24] That is an obvious dilemma.

Some parts of the process have widely been thought as mature, such as configuration management and project tracking [24]. Those are vital conditions for projects due to the distributed nature of OSS development, and thus highly employed. On the other hand, those are also the only ways to find historical data and to understand rationales of the selections, since the level of other documentation is very low [24].

Another problem is that researchers have mostly studied quite large and active projects (e.g. [15]), while most of the OSS projects are small and sluggish [24]. It is recognized that large and active user and developer base usually implies high quality, but this is not possible for all OSS projects — simply, there is not enough developers available in the planet.

While the interest in OSS rises, not all the assumptions related to OSS development are valid anymore. There will be more and more users without technical background, and also the demand of developers increases in quantity, which inevitably leads to decrease in quality of developers. Leaning on most talented professionals is not possible anymore.

2.2 Recommended research and practice

As the commercial usage of OSS increases and non technical users start to employ OSS, quality expectations are much higher than earlier, when end-user was seen as a co-developer tolerating crashes, hunting defects, and fixing the code. The quality assurance must be advanced to development phase before the product is delivered to users. Similarly, as an increasing number of developers are taking part in OSS development, the need for stricter methods and processes becomes evident.

It is commonly believed that achieving high quality SW products requires a precisely defined process to be followed in the production of the SW. The more systematic and manageable the process is, the higher the quality of the outcome [6].

A number of models have been introduced for evaluating, improving and standardizing the SW production process, the most widely known being CMMI and ISO 15504. However, the most prominent SW process improvement models have some deficiencies. In OSS development, the product lifecycle and process that is used to produce the SW is totally different from traditional development models. Thus, generic process improvement models cannot be applied and even traditional, proven quality assurance techniques are challenged by distributed development, dynamic instead of hierarchical development team structures and even cultural issues. Adaptation to the nature and philosophy of OSS development is needed.

The most effective quality assurance techniques are testing and peer reviews. These are included in virtually all modern SW development process models and process improvement models in some form, but typically not suitable for OSS development as is. However, as a start point for adapting peer review techniques into OSS development, there is, for

example, a variety of modified inspection processes from lightweight to strict ones (e.g. [21]), and tools to support geographically dispersed teams [12].

In the area of testing, more attention to test coverage must be paid. There should be adequate test plans, using of which ensures that most of the defects are caught before the SW is delivered to an end-user. Also regression testing should be used more widely, since OSS development is very much evolutionary in nature. Test driven development [1] could be used to produce and execute test cases. As a code-centric method, it is well suited for OSS development.

The main challenge in OSS quality assurance is commitment, since conventional OSS developer is not paid or has no formal authorities. Typically quality assurance has been thought as a boring job to do, and developers tend to pick up tasks where they can utilize their creativity. When commercial firms are involving in OSS development with their high quality requirements, it is obvious that there will be some developers who get paid to perform source code reviews and extensive tests.

In traditional SW development, adding developers to late SW project makes them later [4]. OSS projects have proven that quality assurance related tasks, such as reviewing, testing and locating defects, can be parallelized [20]. Thus, the more reviewers and testers a project has, the higher quality it will produce. However, commercially viable applications targeted for the masses cannot rely on end-users' expertise on that area. The quality assurance has to be implemented as part of the actual OSS development process by adopting established quality assurance techniques fitted for the OSS philosophy.

3. Usability in OSS Development

Usability is an important quality characteristic of a SW products and systems. The importance of usability has been emphasized especially in HCI literature, which is next discussed and related to the OSS context. The importance of usability has been emphasized especially in HCI literature, which is next discussed to introduce the central concepts, and afterwards related to the OSS context.

3.1 Recommended research and practice

Currently a widely accepted definition of usability defines it as ’the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use’ [14]. Related to the definition, important is to acknowledge that usability is always related to specific users, and to their specific goals and tasks that are done in a specific context of use. User-centered design (UCD), on the other hand, is an approach to interactive system development focusing specifically on making systems usable [14]. In the standard, UCD is characterized by its goal of producing usable systems, and by the principles of active user participation, appropriate allocation of functions between users and technology, iterative design and multi-disciplinary design. [14] In addition, the standard defines a set of UCD activities that include planning the UCD process, understanding and specifying the context of use, specifying the user requirements, producing design solutions, and evaluating the designs against the requirements [14].

However, UCD is a very vague concept with a multiplicity of meanings attached to it. It has been analyzed as a multi-dimensional concept, revealing a number of dimensions of user centeredness (i.e. as user focus, work-centered, user participation and system personalization, [13]). Different UCD methodologies acknowledge these dimensions to a different extent. Some of the methodologies emphasize user focus. An ideal is that the focus is on each individual user, but typically the focus is limited to typical, average or even fictive user (see e.g. [5]). Other methodologies, instead, emphasize users’ work practices and tasks as the main point to first understand in depth, and afterwards to carefully redesign (see e.g. [3]). In some methodologies, furthermore, active and direct user participation has particularly been emphasized (see e.g. [11]). Despite the differences, all these methodologies emphasize the importance of understanding the user, his/her tasks or work practices and the context of use. Based on the understanding, the tasks or work practices are to be carefully redesigned. User involvement in the design process is argued for. User feedback should be sought early, and the design solution should be iterated based on the user feedback.

3.2 Current research and practice

In the OSS development literature usability and UCD have been discussed to some extent. However, there are many controversies and limitations in the ways these constructs have been addressed in the literature to date. In the literature search, we found seven published OSS papers dealing with these issues. We analyzed these papers to see how they approached usability and UCD, and what kind of recommendations they give to improve usability in OSS development.

First of all, none of the papers offered any clear definition of usability or UCD. However, particularly user involvement has been emphasized as an important element of OSS development. Nevertheless, in OSS development the distinction between user and developer is blurred; typically the developers have also been the users of the OSS. For that reason, however,

OSS development has been argued of utilizing a truly 'user-driven' approach [17, 23]. Nevertheless, the OSS community is now starting to acknowledge that 'we are not our users' [10]. People are starting to acknowledge that from the point of view of naïve, 'typical', non computer professional users, usability of the OSS is poor, and the development process should be characterized as anything but 'user centered' [2, 7, 17, 19, 22, 23]. Therefore, usability is becoming a relevant topic of research in the OSS context, even though, to date, it has not been examined much.

The field of HCI emphasizes the need of trained usability specialists to contribute to the development. The users can not act as usability specialists, because they are not trained for developing and ensuring usability, even thought they encounter the usability problems while using the system. However, problematic is that usability specialists do not typically participate in the OSS development, and the OSS developers do not have the knowledge and skills needed. In addition, typically there are no resources for UCD in OSS development. [2, 10, 17, 22, 23]. Furthermore, if usability is acknowledged, this typically happens too late in the process. Finally, no UCD methodology is typically employed, because this can be seen as being in contrast with the 'open source philosophy'; it is assumed that in OSS development there is no possibility for systematic UCD or formal process models. [2, 10, 17, 22, 23]

However, it is argued that there is a great potential for usability specialists to start to contribute to OSS development. The papers recommend bringing usability specialists to OS projects and use of expert based evaluation methods [2, 17, 22, 23]. Another solution suggested is the use of usability guidelines that outline the best practices of HCI design [2, 17]. Large corporations that nowadays participate in OSS development can provide both professional usability resources and HCI guidelines for OSS development [2]. Furthermore, automatic usability testing and bug reporting are suggested as solutions. Typically there is a large user base and existing procedures for bug reporting. [2, 22, 23] Also empirical usability testing is argued for [17, 22, 23]. However, a noteworthy observation is that a lot of UCD methods that don't include real users at all are recommended. Finally, an interesting issue to be considered is distributed UCD for the OSS development context. In HCI, support for this type of work has not been discussed much [17]. All in all, OSS development is a new, challenging context to the HCI community to enter into.

4. Concluding discussion

This paper has reviewed literature on quality and usability assurance in OSS development. The main results are summarized in table 1.

Table 1. Quality and usability in open source software development.

	Quality	**Usability**
Current practice	Developers are very talented, lead developer checks patches, users assumed to report bugs	Users assumed to report bugs and to be the co-developers with technical background
Current research	Status quo reports, surveys on current quality techniques in OSS development	Very few articles published, usability and UCD very vaguely defined
Recommended practice	Shift to stricter quality assurance methods and processes as commercial interest increases and more and more developers are participating in development, produce plans and documents to support communication, use inspections and reviews, pay more attention in test coverage, test-driven development and testing performed by developers, not users	Understand and specify the user, his/her work practice/tasks and the context of use, and carefully redesign the work practice/tasks based on the understanding, actively involve the user, gather early user feedback and iterate the design solution based on the user feedback
Recommended research	How to adapt traditional quality assurance methods, such as inspections, for distributed development and to fit the OSS philosophy?	How to adapt traditional UCD methodologies for distributed development and to fit the OSS philosophy?

Typically OSS research tries to recognize silver bullets of stunningly successful OSS development in order to revolutionize traditional software engineering. We took an opposite approach. In our viewpoint, as commercial interest rises, as non-volunteer and non-distinguishable-talented developers are taking part in OSS development, as even more non-technical users are entering into OSS world, the cornerstones of OSS will break down. To ensure high quality and usability in that situation, proven methods and processes must be adapted to OSS philosophy and introduced into the development.

Regarding practical implications, we emphasize the gap that seems to exist between the current practice

and the recommended practice outlined in table 1. Clearly, there is work to be done to make usability and quality assurance as normal part of OSS development. Regarding limitations and paths for future work, clearly more empirical research in OSS development context is needed - particularly qualitative empirical research aiming at understanding in depth the current ways and challenges involved with usability and quality assurance in OSS development. Based on that understanding, one might start to consider the adaptation of the existing SE and HCI methods and processes to OSS philosophy and to the distributed environment of the OSS development.

References

[1] Beck, K. (2002): Test Drive Development: By Example, Addison-Wesley. ISBN 0-3211-4653-0.

[2] Benson, C., Müller-Prove, M., Mzourek, J. (2004): "Professional usability in open source projects: GNOME, OpenOffice.org, NetBeans." Extended Abstracts of the CHI2004. Pp. 1083-1084.

[3] Beyer, H., Holtzblatt, K., Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann Publishers Inc, San Francisco, 1998

[4] Brooks, F. P. (1975): The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley Publishing Company.

[5] Cooper, A., The Inmates Are Running the Asylum: Why high-tech products drive us crazy and how to restore sanity, Macmillan, Indianapolis, 1999

[6] Deming, W.E. (1982): Quality, Productivity and the Competitive Position, MIT Press, Mass.

[7] Feller, J. & Fitzgerald, B. (2000): A Framework Analysis Of The Open Source Development Paradigm. In the Proc. of 21st International Conference on Information Systems, December 10-13, 2000, Brisbane, Australia. Pp. 58-69.

[8] Fitzgerald, B. (2006): The Transformation of Open Source Software. MISQ 30(3). Pp. 587-598.

[9] Fitzgerald, B. & Ågerfalk , P.J. (2005) The Mysteries of Open Source Software: Black and White and Red All Over? Proc. of the 38th Hawaii International Conference on System Sciences, IEEE Computer Society Press.

[10] Frishberg, N., Dirks, A. M., Benson, C., Nickel, S. & Smith, S. (2002): Getting to know you: open source development meets usability. Extended Abstracts of the CHI 2002. Pp. 932-933.

[11] Greenbaum, J. and Kyng, M. (eds.), Design at Work. Cooperative Design of Computer Systems, Lawrence Erlbaum Associates, New Jersey, 1991

[12] Hedberg H (2004): Introducing the Next Generation of Software Inspection Tools. Proc. of the 5th International Conference of Product Focused Software Process Improvement, pp. 34-247.

[13] Iivari, J. & Iivari, N. (2006): Varieties of User-Centeredness. Proc. of the 39th Annual Hawaii International Conference on System Sciences. IEEE Computer Society Press.

[14] ISO 13407, Human-centered design processes for interactive systems. International Standard, 1999

[15] Mockus, A., Fielding, R. & Herbsleb, J. (2000): A Case Study of Open Source Software Development: The Apache Server. Proc. of ICSE 2000, pp. 263-272.

[16] Nichols, D., Thomson, K. & Yeates, S. (2001): Usability and Open Source Software Development. In the Proc. of the Symposium on Computer Human Interaction. Pp. 49-54.

[17] Nichols, D. & Twidale, M. (2006): Usability Processes in Open Source Projects. Software Process Improvement and Practice 11. Pp. 149-162.

[18] Niederman, F., Davis, A. Greiner, M., Wynn, D. & York, P. (2006): A Research Agenda for Studying Open Source I: A Multilevel Framework. Communication of the Association for Information Systems 18. pp. 19-149.

[19] Pemberton, S. (2004): Scratching Someone Else's Itch (Why Open Source Can't Do Usability). Interactions January + February. P. 72.

[20] Raymond (1999): The Cathedral & the Bazaar: Musing on Linux and Open Source by an Accidental Revolutionary, O'Reilly & Assoc.

[21] Tervonen, I. Iisakka, J. & Harjumaa, L. (1998): Software Inspection - A Blend of Discipline and Flexibility. Kusters R., Cowderoy A., Heemstra F., and Trienekens J., (ed.): Project Control for 2000 and Beyond, Proc. of ENCRESS-98, Shaker Publishing B.V., 1998, pp. 157-166.

[22] Zhao, L. & Deek, F. (2005): Improving Open Source Software Usability. Proc. of the 11th Americas Conference on Information Systems. Pp. 923-928.

[23] Zhao, L. & Deek, F. (2006): Exploratory inspection: a learning model for improving open source software usability. Extended Abstracts of the CHI2006. Pp. 1589-1594.

[24] Zhao, L. & Elbaum, S. (2003): Quality assurance under the open source development model. Journal of Systems and Software 66(1). Pp. 65-75.