

An Analysis of Involvement of HCI Experts in Distributed Software Development: Practical Issues

Görkem Çetin¹, Damiano Verzulli², and Sandra Frings³

¹ Gebze Institute of Technology
gcetin@gyte.edu.tr

² CINECA
damiano@verzulli.it

³ University of Stuttgart
sandra.frings@iao.fraunhofer.de

Abstract. Traditionally, free and open source software (F/OSS) developers have focused more on the features of a specific application, most of the time ignoring the necessity of user-centric design. This has mainly stemmed from the fact that developers have little interaction with HCI studies, knowledge bases and reports. Moreover, the lack of user interface designers has resulted in a lack of awareness of this area. As a consequence, the user centric design phenomenon within F/OSS applications has been neglected. In this paper, we have mentioned various problems that would slow down a F/OSS project development towards a user-engineered software, and investigated the ways that HCI experts and developers interact with each other and researched bug reporting systems by means of eligibility to issue usability bugs. For the conclusion part, we have explored possible ways to achieve a user-centric design in a project with asynchronous interaction among geographically distributed developers.

Keywords: F/OSS, HCI, usability, open source, distributed usability, user centric design.

1 Introduction

Ignoring the necessity of user centric design [2] by the developers of the same project has been a global problem of project managers since the very beginning of software development paradigm. There has been a notorious conflict between the developer seeking for an extended ability bundled with paranormal features of the software and the project manager trying to convince the developer to build a framework exploiting user centric development methodologies. For a long time, the project managers and the developers of many open source projects have been pointing to the same person, so this problem has been solved in a manner that it has never existing. As a result, user centric design phenomenon within F/OSS applications has been neglected [3]. Starting from the beginning of 2000, commercialization of open source software has opened the possibility for the masses to realize the effects of “an open project does not mean it should suffer from usability” paradigm. End users, developers and

customers have become the main sources of reporting usability bugs. However, there's still room for HCI experts to take part in this table as the fourth player.

Problems above can be seen as a consequence of the particular history of the F/OSS community. Right after the birth of the F/OSS community, in 1983¹, all the developer efforts were spent for the development of technical applications targeting computer scientists. At that time there was no previous F/OSS applications and the development of the building-blocks (editor, compiler, linker, debugger, etc.) was necessary as the first step. Once becoming production-ready, these building-blocks could have acted as enablers for a much wider community of programmers that, with such tools, would have been able to develop many other kinds of applications.

Although there have been some attempts to simplify the usage of several F/OSS applications, nothing serious happened until 1996. At that time end-user personal computers were running a proprietary operating system together with a certain number of proprietary applications. Being driven by marketing needs, software corporations developed such applications focusing on several factors, including usability: “...*the effectiveness, efficiency and satisfaction with which users can achieve tasks...*” by the way of such operating system and applications [4].

It was in this timeframe and context that something new happened within the F/OSS community. Following the announcement of the KDE project (1996) and the GNOME project (1997), F/OSS community started taking care of end-users needs and, as such, building two completely new desktop environments. Both KDE and GNOME had the end-users needs within their mission remit, recognising usability as a critical factor from the beginning of both projects.

Comparing the way usability norm was approached by proprietary vendors and by the F/OSS community, it is worth noting that, due to the commercial nature of proprietary software vendors, usability was (and still is) seen as a revenue generator: the more usable the software, the more licenses sold, the more revenues generated. This issue can heavily impact the development process as companies, being driven by the business needs, will force the adoption of usability requirements even if they are often underestimated by many programmers due to their technical mindset. Within the F/OSS community things are much different [5] as the main driver for the developing activities is the “freedom” of the movement itself. A freedom that, as such, does not imply and/or enforce anything in terms of usability compliance.

Several other factors have had a critical role in taking the F/OSS community to completely ignore, or to heavily limit, the necessity of a User Centric Design (UCD). With the launch of the KDE project, for the first time, the F/OSS community focused on end-users. Communications with such users, the understanding of their needs, their approach to computer usage and lots of other issues were completely new problems that the F/OSS community needed to face. Moreover, while since the early days the F/OSS community succeeded relying only on its internal competences, with the advent of F/OSS desktop computing the situation quickly changed and a completely new set of skills was needed. Interface design, visual communication and information

¹ Here we identify the F/OSS community with the GNU project, as this was the first attempt to formally develop a “free” operating environment. The GNU project has been officially started by Richard Stallman on Sep 27th 1983, with the announcement stated on <http://www.gnu.org/gnu/initial-announcement.html>

architecture became mandatory concepts for a winning desktop environment. Those concepts were mostly unknown and unavailable within the F/OSS community member [6].

Furthermore, if it were easy to put F/OSS programmer around the same table discussing about applications technical details, the rising of desktop F/OSS required the establishment of new links between F/OSS community and HCI community. Actually, it was (and still is) hard to establish such new links as both communities grow among different roots.

Based on what have been written above, we think that to improve the usability level of F/OSS applications, and specifically, to increase the current involvement of HCI experts in whole development process, several actions need to be jointly adopted. We refer to the need, for the F/OSS community and HCI experts' community, to create a distributed development, involvement, evaluation and reporting methodology for every F/OSS usability issues. It's also fundamental that usability requirements needs to be taken in proper account since the very beginning of every F/OSS projects: all programmers know that application requirements need to be formally defined before the start of the developing activity; in the same way usability requirements need to be specified and defined well before the development activity. Finally, the need to maintain or even increase the current trend in desktop F/OSS development will surely impact the usability of future F/OSS applications: the development trend of projects like KDE and GNOME follows a very tight time-frame, with improvements, also in usability area, that are much visible in respect to proprietary counterparts. Should such progress continue for the future releases, it is easy to expect a very comfortable and usable future F/OSS desktops.

2 Interaction Issues

In a distributed environment, limited and asynchronous information flow leads to a problem of low performance among developers, compared to collocated teams. In such a team, most of the communication is ideally conducted electronically (e-mails, phone, teleconferencing, emails, etc) – sometimes all team members meet in a predefined location with changing periods. For example, most KDE developers meet at least twice in a year, but some less technical and low-profile projects' developers may find it unnecessary and expensive even to meet annually. The Internet can both be a viable and a problematic way of collaboration, since it has a dull side which limits vocal peer conversation. However F/OSS developers can benefit from distributed intelligence where there is always room for usability experts. Low-budget projects can also have an opportunity to find usability people, and merge them into their projects to increase the usability profile of the resulting product [7].

Reitmayr and Mühling [8] define three basic problems that would slow down the development towards a user-engineered software:

1. The basis of usability work, a clear definition of the target users, their tasks and their requirements are often missing.
2. Missing hierarchical decision paths may pose a problem in larger projects where a usability expert needs to convince each relevant developer instead of the head of department.

3. A traditional focus on technical rather than GUI-related issues may require a major redesign of the information architecture and interaction design of software.

It's convenient to add the following items about why usability experts face problems while taking an active part in F/OSS development.

1. With usability experts taking an active part in the project, the time-to-market is slowed down.
2. A developer should cope with other roles – now he is not the only decision maker.
3. The developer without knowing its user base assumes that the requirements for the new application are the same as the requirements in all other applications.
4. The necessity of a user interface developer is questioned after introduction, adoption and exploitation of interface design software by non-designers.

As we have stated before, the fact of problematic way of having collaboration leads to a low profile of interaction modalities. Bug reports may be well hidden from the project volunteers, the documentation subsystem may not benefit from the modern application development interfaces, or the lack of consistent and coherent software including a scheduling tool, task list or a trouble ticket system may yield to a frustration of the HCI expert. Vast variety of tools geared towards high tech developers, always questioned efficiency of high volume mailing list traffic, and a lack of initiative to amend the requests of the usability expert pose a diverse working environment than a model where the usability expert joins a project from the very beginning, sets the requirements, initiates a user centric design under project manager's acknowledgement, conducts necessary usability tests with predefined tasks matching the focus user group.

3 Focusing on Usability Reporting Tools

Bug reporting tools vary with the way they work. While some utilities (i.e. crash reporting tools) involve little end user activity by only sending proper crash debugging information, others can request a plea of subjective and objective input from the end user. A few applications, when clicked on a specific menu item, forward the user to the application development web page and requests to answer the qualitative and quantitative questions asking the degree of users' satisfaction [9].

There are some prototypes [10] to report usability bugs, having a major redesign of a generic user reporting tools and real world examples which has a working backend to support interested developers [11]. Communication channels between the end user and developer is limited, so application-bundled can be an effective solution. However, in free software world, application developers tend to rely on web based bug reporting tools like Bugzilla and Mantis, ignoring the necessity to bundle their application with a consistent feedback tool. While this approach is changing over the recent years with more applications return feedback from users, we believe that the ignorance of developing a bug reporting interface which collects not only textual data but hypermedia from users seldomly makes it complicated to exchange proper information among developer and the reporter.

Current tools are not convenient for reporting usability issues with the following reasons:

- They don't have a mechanism to interactively record, upload, show, maintain and comment on user submitted videos, images and voice.
- There's no way to merge a note to an attachment to show the submitters' and developers' opinion, annoyance and feedback. Trying to spot a minor usability issue may not be explained verbally, and hence needs a graphical representation. Unfortunately, not all computers are bundled with a painting and drawing application.
- Current bug reporting tools have an increased complexity which is trying to spot all kinds of problems on the direction of the developer, ignoring the mental model of the user.
- An average bug reporting tool requires to fill a considerable amount of information, some of which are not immediately pertinent to the end user or HCI expert, making it sometimes impossible to submit a bug thus leave the HCI expert out of the scope of the project.

Under the light of the facts above, the lack of a suitable usability reporting interface results in some issues. First, number of reporters and reports thereof decreases. Most of the critical bugs never go into the bug database, rendering it unusual to increase the quality of the code. Second, usability reports are handled by the mailing lists and forums instead of a database, which is hard to follow, fix and give proper feedback to the reporter. And finally, since the aim of the bug reporting tool is often misunderstood, the end user (sometimes the usability expert) starts to discuss about an issue and/or report a problem he cannot solve, mostly ending up with closing the bug because of misusing the bug reporting tool.

4 Analyzing Bug Reports

There has been related empirical researches showing findings in F/OSS development community [12], however there's a lack of researching how developers really exploit the presence of HCI experts and usability bugs. In this study, KDE (K Desktop Environment) project has been identified as core materials to be investigated. 100 random samples have been taken from KDE, and usability and non-usability bugs have been identified and classified according to their severity. For a side by side study, a cross-comparison can be practiced with the bug database of a Linux distribution to see the differences in usability bugs with respect to maturity of the project, development phase, number of developers, number of active HCI experts, and awareness of the project. This comparison will also identify the main deviations of bug results of a desktop OS vs. desktop applications.

The following considerations and assumptions are taken during the analysis of bug database.

1. File attachments are not counted as another thread item
2. Application crashes are not counted as usability bugs
3. Number of threads also includes the first post
4. Duplicate information in the thread is not counted as another thread item

Table 1. KDE bugs analysis. The bug numbers are randomly taken from bugs.kde.org between years 2001-2005.

Heading level	Total number	Average number of threads	Number of closed bugs (rate)
All bugs	100	2,75	80 (80%)
Usability bugs	27	2,78	27 (100%)
Non-usability bugs	73	2,74	53 (73%)

From this table, we can immediately see that roughly a quarter of all bugs are usability related. While the detailed investigation of the bug reports haven't been carried, it still remains a question whether a bug reporting tool with usability extensions mentioned in chapter 3 would:

- Increase the rate of usability bugs compared to non-usability bugs
- Increase the average number of threads because more people can be involved in the discussion since the report is more comprehensive and hypertext-based.
- Decrease the average number of the threads since the reports are more meaningful, thus eliminating the repetitive questions from developers asking for clarification of the issue.

5 Behaviour Patterns

Lack of HCI experts have always been a delaying and eroding factor of usability paradigm in F/OSS projects. The basic rule of thumb of "far away syndrome" where the developers cooperate with an asynchronous collaboration framework also hits the usually non-technical HCI experts, resulting in an alienation to the project. Up to now, many papers have investigated the hybrid form of developing and implementing F/OSS software, and also identified several key factors shaping the collaboration between the developers in a community [13] or defined success measures of F/OSS projects [14], however none of them had considered the usability experts' involvement and behaviour patterns in F/OSS projects and the relationship between the project success and the degree of user centric design in the project.

The time at which a usability expert becomes involved in a F/OSS project is critical. Early involvement offers a better chance of a strong influence on the product user interface, since the acceptability of the interface designer will likely be higher within the project. In this way, a paper prototyping is possible, allowing continuous amendment cycles during the product design stage. Moreover, the usability expert will not only develop user requirements and develop user profiles, but he will also be able to consistently fix and/or offer usability bugs found within the product at an early stage. The table below shows different involvement stages of an usability expert in a F/OSS project and their outcomes [15].

This analysis shows that if the HCI involvement is high, then influence on the product interface, acceptance of the expert in the community, and applicability of the paper prototyping is also high. For the late involvement scenario, the focus point of

Table 2. A table showing the degree of sustainability and progress and degree of mutual interaction. Communication hub refers to various collaborative media (i.e forums, mailing lists, bugzilla interface) project stakeholders are involved in.

Discussion model	Characteristic of usability discussions	Possible results
Between-developer	<ul style="list-style-type: none"> • Subject to lose focus in time • Limited to mutual understanding of a concept with limited knowledge about HCI • Usually via mailing lists or instant messaging environments • Short living discussion • No potential mechanism to maintain and sustain the discussion • Subjective 	Likely yields to a successful feedbacking of other project developers if the mutual mailings are carried on at mailing lists
Between HCI expert and developer	<ul style="list-style-type: none"> • Usually initiated and directed by the expert • Progression is directly related to expert's behaviour as well as the complexity of request • Seldomly yields to an initiative to start a UI refactoring • Perceived usability is generally a dominant feedback context 	The expert should convince the developer
Between HCI expert and communication hub	<ul style="list-style-type: none"> • Qualitative as well as quantitative discussions • Well maintained, generally long lasting • Yields to providing reports • Objective and methodological rather than depending on personal ideas 	Generally yielding a far more better result, however this model is not much widely exploited

Table 3. Different involvement stages and their outcomes

Discussion model	Early involvement	Involvement halfway	on Late involvement
Influence on the product user interface (UI)	High	Moderate	Less likely
Acceptance of the expert	High	Moderate	Moderate
Applicability of paper prototyping	For all UI elements	For the upcoming UI elements	Almost none
Focus point of usability expert	Developing user requirements	Amendment of UI	Crucial usability bug fixing

the usability expert narrows to crucial usability bug fixing. Unfortunately, we have found evidences that the acceptance of the HCI expert in the community considerably lowers in the case of late involvement.

6 Results and Discussion

In this paper, we focused on the involvement and the adoption of usability experts into F/OSS projects. We analysed the problems and how to overcome them in a straightforward manner. One way to get usability experts into F/OSS projects is to make developers more aware of basic usability principles. The outcome of this action can be stated as follows:

1. Developers will be able to evaluate their own projects, which in turn yields better quality and usable software.
2. There will be a mutual understanding of two groups (i.e. developers and interaction designers).
3. A usability expert will show the user's perspectives and focus on user profiles, finely defining the target groups.

We see the absence of key components like usability reports, usability laboratories, usability experts and companies specialising in usability for F/OSS, to be taken in appropriate consideration during all of the development process. Moreover, there's a lack of usability bug reporting tool which can be used to submit, store, modify and maintain user submitted videos, audio files and pictures showing the usability issues on a particular software UI.

For a radical change of the current status, not only it is necessary to take into account usability standards requirements, but it is critical to adopt them as early as possible in the development process. Such an approach, in effect, can be seen like an adaptation and/or a simplification of the directives specified by current proprietary standards provided by external organizations [16].

Acknowledgments. We would like to thank the KDE developers for providing their Bugzilla data for this study to analyze their bug database

References

1. Nichols, D.M., Twidale, M.B: The usability of Open Source, First Monday issue 8.1 (2003)
2. Frans E.: Open source usability is a technical problem we can solve our own, reached at <http://www.newsforge.com/article.pl?sid=04/07/07/1640244>
3. Muehling, J., Reitmayr, E.: Integrating Usability with Open Source Software Development: Case Studies from the Initiative OpenUsability: tOSSad OSS 2006 Workshop proceedings, pp. 65 (2006)
4. ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability

5. Meeting the challenge of open source software usability, by Benson, C.: British HCI Group – (Autumn 2004) (Interfaces 60) – <http://www.bcs-hci.org.uk/interfaces/interfaces60.pdf>
6. OpenUsability.org: Usability and Open Source Software, by Muehlig, J., Paul, C.L.: British HCI Group – (Spring 2006) (Interfaces 66) <http://www.bcs-hci.org.uk/interfaces/interfaces66.pdf>
7. Çetin, G., Frings, S., Verzulli, D., Jovanovic, U.: Usability involvement in F/OSS projects, a usability guideline. tOSSad project report, funded by EU FP6-IST3 contract no 015981.
8. Muehling, J., Reitmayr, E.: Integrating Usability with Open Source Software Development: Case Studies from the Initiative OpenUsability: tOSSad OSS 2006 Workshop proceedings, pp. 65 (2006)
9. OpenOffice.org office software web page, reached at <http://www.openoffice.org>
10. Nichols, D.M., McKay, D., Twidale, M.B: Participatory Usability: supporting proactive users, pp. 4
11. Likeback feedback software, reached at <http://basket.kde.org/likeback.php>
12. Sandusky, R.J., Gasser, L., Ripoche, G.: Bug Report Networks: Varieties, Strategies, and Impacts in a F/OSS Development Community
13. Lin, Y.: Hybrid innovation: The dynamics of collaboration between the FLOSS community and corporations: Journal of Knowledge, Technology and Policy, vol. 18(4) (Winter 2006)
14. Crowston, K., Annabi, H., Howison, J., Masango, C.: Towards A Portfolio of FLOSS Project Success Measures. In: Collaboration, Conflict and Control: The 4th Workshop on Open Source Software Engineering, International Conference on Software Engineering (ICSE 2004), Edinburgh, Scotland (May 25, 2004)
15. Çetin, G., Frings, S., Verzulli, D., Jovanovic, U: Usability involvement in F/OSS projects, a usability guideline. tOSSad project report, funded by EU FP6-IST3 contract no 015981
16. Jokela, T., Iivari, N., Matero, J., Karukka, M: The Standard of User-Centred Design and the Standard Definition of Usability: Analyzing ISO 13407 against ISO 9241-11.