A Review of Tool Support for User-Related Communication in FLOSS Development

Aapo Rantalainen, Henrik Hedberg, and Netta Iivari

Department of Information Processing Science, University of Oulu

Abstract. Free/Libre/Open Source Software (FLOSS) projects rely on Internet tools for communication and in coordinating their work. Communication between developers is well supported in FLOSS projects, but user-developer communication has proven out to be challenging. This paper examines the following questions: "What kinds of means for communication exist in FLOSS projects for user-developer communication? What kinds of means should there be?" We have carried out a literature review addressing communication in FLOSS projects, and contrasted the findings with Human-Computer Interaction (HCI) literature on user-developer communication. HCI literature indicates that user-developer communication is needed during requirements construction, design and evaluation tasks, and HCI specialists are needed for orchestrating the communication and the user related tasks. Communication during the evaluation task is somewhat supported in FLOSS projects, but design and requirements construction are badly in need for support, even though ideas have already been presented. In addition, HCI specialists are in need of different kinds of communication support in FLOSS projects.

Keywords: Free/Libre/Open Source Software, Human-Computer Interaction, User-Developer Communication, Tool Support.

1 Introduction

The term FLOSS means Free/Libre/Open Source Software. It refers to free software, defined by Free Software Foundation [28] and open source programs defined by Open Source Initiative [51]. This software is determined by licenses. If a license allows users to modify and share software, it is considered as free software. Practically this means users must have access to the source code, thus the name open source. But open source is more than an available source code, it is a philosophy of studying the code, improving it, customizing it, and sharing it [16]. However, free software is also an ideology and a lifestyle [65, 68].

FLOSS projects are usually described by concentric circles [19]. The most inner circle represents coders who have direct write access to the official code. The next circle represents coders who send patches. Last circles are users, first active ones and then passive. Working is very meritocratic [64]. Contributors who have distinguished themselves by the quality of their work are invited to join the inner circle and gain more responsibility in the project [21]. FLOSS development

S.A. Hissam et al. (Eds.): OSS 2011, IFIP AICT 365, pp. 90–105, 2011.

originates in the hacker culture with highly talented developers [41, 42, 63], but recently also non-technical users have found FLOSS solutions; e.g., Linux operating system, OpenOffice.org application suite and Mozilla Firefox web browser are widely known and used by non-technical users. Therefore, in FLOSS projects nowadays there are typically different stakeholders with different perspectives, emphasizing, e.g., end user-oriented aspects or technical implementation-specific details. The onion model describes only coders, not e.g. translators, manual writers, HCI experts nor any domain specialist [33]. Communication between these different stakeholder groups and perspectives is a challenge [33]. Especially it has been noticed that HCI specialists tend to remain outsiders and they do not have any real decision making power in FLOSS development [1, 9, 12, 15, 49, 61, 62].

This paper is focused on software targeted to the end users (i.e. not system libraries) and will examine: "What kinds of means for communication exist in FLOSS projects for user-developer communication? What kinds of means should there be?" Communication, generally, refers to the successful exchange of information so that the sender and receiver understand each other [14]. We will examine user-developer communication in FLOSS development by relying on ideas from HCI research on user-developer communication. This literature is contrasted with the existing research on communication in FLOSS development. We will critically review the means already suggested for user-developer communication.

The article is structured as follows. The next two sections cover a general introduction to the communication means used in FLOSS projects as well as the existing means used for user-developer communication. The fourth section presents guidelines from HCI research on user-developer communication, and the fifth section contrasts those with the findings related to communication in FLOSS projects. A number of areas for improvement and associated paths for future work are identified.

2 Existing Communication Means in FLOSS Projects

Next, few common characteristics of communication in FLOSS projects are described, even though it is acknowledged that there is no particular, explicitly defined FLOSS development methodology or model [46]. Although the open source development process is not well defined, some general characteristics can still be identified. Open-source projects attempt to ship out minimally working prototypes at the earliest possible time [13]. The most well known principle is "release early and release often" [54].

Almost always FLOSS projects begin by one developer scratching a personal itch [25, 54]. Thus software might not be even planned to be used by anybody else [58]. Typically a FLOSS project is directed and managed by the initial creator and the requirements come from the programmers themselves. Massey gives additional two sources of requirements: users and standards [43].

FLOSS projects have some common practices for communication, but they have also unique practices [5, 21], as well as unique structures [20]. They are

project oriented, not organization oriented like closed source projects [5]. Because FLOSS projects are often distributed [64], there is a need for methods which support distributed development. Actually, there is a long list of existing communication means used in FLOSS projects that indeed support distributed development. These means are 1) email and mailing lists, 2) web sites, 3) version control systems, 4) bug trackers, 5) real time chats, 6) wikis and 7) web forums. [5, 26].

Often projects use only emails and mailing lists [21]. The most important benefit of them is time independency. Developers can live different continents and different timezones [21]. Email communication is sufficient, so it is used. Fogel claims that a mailinglist is the most used communication channel in FLOSS projects [26]. This kind of an open mailinglist is not scalable, however. If software has millions of users and even a small part of them are posting, the input rate is still bigger than any reader can handle. Therefore there must be dedicated lists. [26].

Web sites of FLOSS projects can be one-way channels for bigger audiences [26]. Not all projects even have other web sites than the bug tracker. Web portals are virtual workspaces that merge several channels to a web site [31]. Ankolekar and colleagues claim that project's websites often have relatively little to offer to non-technical users of the software [2].

Wiki is fast, but it isn't a realtime communication channel. The challenges of Wiki include cohesion of totality and duplicate working [26]. Challenges can be minimized and quality will be better, if there are both implicit (e.g. rules and guidelines) and explicit (e.g. discussion pages) coordination [39].

A web forum can be just an interface to a mailinglist, or it can be meant only for a browser. Web forums are categorized and threaded by default so they can handle more traffic than mailinglists. Forums are typically focused for users. [26].

Bug trackers, bug repositories or issue trackers are places to store found issues and problems. They can be part of the broader web site. They are not meant for discussions [26], but can be used for coordinating and planning releases [26]. Bach and colleagues criticized that bug tracker is a hidden place to do development [5]. The amount of tickets in a bug tracker is of no value in itself. One empirical study showed that 36 % of Eclipse bugs were invalid or duplicates [3].

Mockup pictures are easier to understand than source code, due to which there can be more people to discuss and then also unwanted or unhelpful postings from people who do not understand the big picture [5, 26]. As Parkinson said in 1957, which has now come to be known as Parkinson's Law of Triviality, "the more trivial the topic the more conversation" [52].

3 User-Developer Communication Means in FLOSS Projects

"Writing code isn't the problem, understanding the problem is the problem." [22]. One of the main problems in software projects is the lack of domain knowledge [22]. This source of error is eliminated in self-driven FLOSS projects where

domain experts write software for their own needs [45]. But when developers are not users of the software, they need domain expertise [45]. Complex software project's design problems can't be solved by individuals or by homogeneous groups [24]. HCI experts are needed to steer user research studies such as surveys and interviews [53]. Communication is critical to these processes [50]. Nevertheless, "the role of the average user in FLOSS is not clear" [6].

Existing literature has indicated that there is a multitude of tools available in FLOSS projects for user-developer communication (see section 2): users can deliver feature requests and bug reports through these means as well as ask questions, while developers can answer the questions as well as provide user support [29, 40, 60, 69].

However, tools should be chosen based of on the skills of users, not the skills of developers [57]. Despite the existence of these means, it might be that the users are unable to communicate with the developers. They might be unable, unwilling or scared to use these means (i.e. the mailing lists and bug trackers) [9, 15, 48, 70]. "Non-technical users may not have the technical vocabulary valued by developers" [5]. Numerous problems with bug trackers have been listed in connection to reporting usability issues. They enable no recording, uploading, showing, managing or commenting videos, audio or images. Minor usability issues are hard to describe only verbally, and not all users have painting or drawing tools. The existing bug trackers are also too complex and coder centric. They ask many questions that user (or non-technical domain specialist) don't know how to answer. Because of these points not all users are able to use trackers at all. [15].

The skill level of reporters may also vary and they don't know or understand the bug fixing process [38]. Users don't know what is a good bug report either, i.e. what is relevant and needed information [10]. The bug tracker should tell a user what is relevant information and how a user can find it [10], or there should be no open bug reporting at all, but users should use support and feature request forums and moderators should pick up all found bugs into the internal bug tracker [38].

Usability and overall user experience (UX) issues are different than other bugs and issues. Bugs and UX-issues should be segregated, but they both may be visible in the same bug tracker and linked together. [5]. HCI specialists should also be welcomed to a project [4] and they should have an acknowledged role on the project web portal [4] which should have its own tab for UX-issues [5]. 'Confusion reports' and 'surprise reports' are suggested to be used additionally to bug reports. These are reports by active, reflective users and contain descriptions on what the user tried to do and what caused their surprise, whether derived from confusion or not. [8]. Also groups such as 'reference', 'core' and 'bleeding edge' users are brought up. They can be used for initially testing the solution or new features. They can be offered a more advanced tool set for providing feedback, while all users can use some basic tools related to which some automatic summaries are generated to lessen the burden for the developers to analyze

the results. In addition, Open Content projects organized around using certain FLOSS tools are brought up. During them users and developers of the tools collaborate on the task and developers gain feedback. [60].

As mentioned, one problem related to the communication means used in FLOSS projects is that they do not support communicating visually. Related to usability problems and user interface design, textual descriptions are not enough - it may not be sufficient to articulate these issues only textually or it may take too much of effort to do it [15, 48, 49, 61]. Natural language and visualizations should be preferred as well as cooperation during design work supported [33]. In time and location distributed development coders are happy with emails and a version control system, because they are working with a source code, which is text, but domain specialists and HCI experts need also other means of communication, including face to face communication [1, 15, 48, 49].

User interface design by blogs has been mentioned as a way to communicate design solutions in a distributed environment [49, 60]. Dedicated mailing lists for usability discussion have also been brought up [12, 48]. In addition, specific design areas supporting brainstorming and discussion of user related issues have been mentioned [60, 61], as well as a 'usability system' enabling easy to use usability bug reporting with the possibility to use multimedia [15, 49]. Moreover, (remote) usability evaluation and user data gathering tools have been recommended [47–49, 60].

By relying on the traditional HCI literature, the importance of paper prototyping during early design phases has also been highlighted [15]. There should also be user requirements and profiles produced, and users might be interviewed or questionnaires used to inquire them and their needs [15, 60]. It is, however, unclear what kind of communication means there should be supporting these activities. Another problem related to these suggestions is that they seem to assume that the development proceeds similarly like to traditional proprietary software development, with certain phases sequentially following each other, with HCI experts hired to do the job.

All in all, FLOSS has many characteristics, which harness usability work. Typically FLOSS programs, also development tools, are highly modular via plug-ins and customizable via many configurations. This increases complexity related to installing and using them. Documentation is fragmented over forums, personal web pages, and source code. [62]. FLOSS development is rapid and iterative, so rapid that it might look like one code-and-fix attempt from the outside [13], and thus user centric design is challenging [5]. But rapid prototyping can be also good for UX-testing [4].

The lack of coordination causes more frustration to HCI specialists than to coders [15]. Non-hierarchical decision making is not good for HCI specialists, because then it is hard to get their ideas accepted and implemented [15]. The core developers typically make all the decisions related to what to include in the code base, more peripheral developers and users having no decision-making power regarding this [69]. HCI experts may have difficulties in being able to

affect design decisions in FLOSS projects [1, 9, 12, 15, 49, 61]. Coders work for merit, due to which also usability/UX tasks should be made visible and merited, so that these specialists could gain a more authoritative position [5]. Important for HCI specialists is particularly to make themselves known and visible in the project, educate the developers in HCI matters and offer usability feedback [60]. Building trust, providing opportunities to show merit and developing a new workflow with HCI specialists' and developers' work integrated are important for HCI work to become accepted and practiced in FLOSS projects. The new workflow should include a special phase for design, following users suggestion or feedback, preceding actual development. The design should be iterated until finding a satisfactory solution, users' feedback being utilized during the process. [5].

HCI specialists should be acknowledged and empowered by creating another layer of roles into the traditional onion model that is used to depict the decision making structure in FLOSS projects. While the technical stakeholders can be identified as users, contributors, committers and core team members, the human oriented layer introduces positions for non-technical users, usability evaluators, usability designers and a HCI core team, respectively (note that one person can act in several roles). Since the viewpoint to software and produced information differs from source code, also communication means should be different within that layer. The most important communication channel between technical and non-technical project members is shared decision making between the technical and HCI core teams, but it should not be limited to that. [33].

Some FLOSS projects have voting mechanism for bugs and new features that enable users and possible HCI experts to have a say. They differ from project to project. KDE's and some other Bugzilla based bug tracker have "Most hated bugs" and "Most wanted features" and each registered user can give votes to bugs or features (https://bugs.kde.org/). Then there are also brainstorm forums (e.g. http://brainstorm.ubuntu.com/), where users can make any suggestion relating to requirements or design and others can comment and vote for (or against) them. However, these solutions are derived from the FLOSS world, not from HCI oriented research, but they are mentioned here since they somewhat enable HCI specialists and end users to take part in the decision-making process in FLOSS projects.

4 HCI Research Guidelines

In this paper the focus is on supporting user-developer communication, which has been particularly addressed by the field of HCI. Generally one can say that in HCI methods and textbooks the development of interactive systems has been separated into three main phases: 1) requirements construction, 2) producing design solutions, 3) evaluating the design solutions [11, 18, 44, 55]. User-developer communication is needed during all these phases. During the requirements

construction phase, developers need to be in contact with users for the purpose of understanding the users, their needs and problems, and their context of use. Typically, this is achieved through face-to-face contact: users are interviewed or observed in their context of use, even though some methods also mention e.g., surveys as one possibility to inquire users and their needs [11, 18, 44, 55].

During the design phase, different kinds of design solutions are produced for human-computer interaction. Developers should initially carefully redesign users tasks or work practices, before considering software or user interface design [11, 18, 44, 55]. Part of the literature highlights the importance of user contact also during this phase: users may be invited as design partners to produce the HCI solutions together with developers. Typically also this is assumed to take place in a co-located setting, utilizing representations such as scenarios or storyboards to capture the design ideas [11, 18, 44, 55].

One should start the evaluation as early as possible. Typically this entails the use of low-tech prototypes and such, which can be produced as well as modified very fast and easily. On the other hand, it is also important to evaluate the finished or almost finished solution and to check whether it is ready for release. The evaluations typically include users as test participants. [44, 55]. Usability testing is the most widely known and used method. Typically it is again carried out in a co-located setting, in a usability laboratory or in a field setting, but also remote usability testing has been brought up in the literature [32]. Studies in the actual use context after the release are also recommended. This should be done in order to improve the next version. Methods such as interviews, observation, surveys or focus groups can be used, implying again user contact but not necessarily a face-to-face one. [44].

HCI literature recommends developers to cooperate with users, and also emphasizes that HCI experts should take part in the development and 'represent the users' in the development [17, 35, 36]. Typically it is recommended that HCI experts take the responsibility of user contact during these different phases: they observe and interview users during the requirements construction, they invite users to take part in the design process and they organize the usability evaluations involving users as test participants. However, HCI experts and users may find it difficult to have any impact on the solution being developed [35, 37]. They both might only be in an informative role, acting as providers of information or in a consultative role commenting on predefined design solutions, when developers alone proceed to make all the decisions [35]. HCI literature, nevertheless, suggests a more authoritative role for users and for HCI experts representing them - the literature maintains that users and HCI experts should be allowed to have a participative role, taking part in the design process and having decision making power regarding the solution [35].

5 Recommended Communication Means for FLOSS Projects

In this chapter our FLOSS related findings are connected to the HCI literature presented. We acknowledge all the different phases associated with interactive

systems development, but we also emphasize that FLOSS development does not follow the waterfall model. These phases can be found from FLOSS development, but they do not appear sequentially, but are intertwined and overlapping. Scacchi argues that requirements construction in FLOSS development does not consist of the traditional steps following each other, but instead the requirements are asserted or implied in a multitude of textual descriptions in the FLOSS environment, the descriptions being discussed, negotiated and made sense of in a continuous, evolving manner. During the process the requirements are condensed, hardened and concentrated, the requirements construction being comingled with design, implementation and testing. Sometimes the requirements may actually be only implicitly produced as a by-product of implementation. [56]. We acknowledge that requirements construction as well as HCI design and HCI evaluation all take place in FLOSS development one way or the other, but within the boundaries of constraints and characterizations mentioned above. We call these tasks rather than phases due to issues mentioned above.

During the requirements construction task developers may ask for and collect information about users and their needs, while users (or HCI experts representing them) may provide information about these to the development. During the HCI design task, all these parties may produce design solutions (ranging from ideas or rough mockups to finished software) and wish to communicate those to others. During the evaluation task, furthermore, developers may ask for and collect feedback, while users (or HCI experts) may provide it to the development. The actual use can not be separated from FLOSS development but instead both continuously take place after the initial release of the software. During it, developers may provide user support while users may ask for it, this being already labeled as a user-centric strength of FLOSS development [40, 69].

Table 1 summarizes the results of our literature analysis on 1) What kinds of means for communication exist in FLOSS projects for user-developer communication? and 2) What kinds of means should there be? The results were derived by contrasting the existing FLOSS literature on user-developer communication with prescriptive HCI literature on user-developer communication. In FLOSS projects these tasks are overlapping and not even explicitly acknowledged.

Therefore, communication during the evaluation task is somewhat supported in FLOSS projects, but design and requirements construction tasks are clearly in need for further support, even though ideas have already been presented. However, we maintain that all these tasks need to be supported future and plan to experiment with certain solutions in the further. FLOSS development, as a special case of distributed software development, is very tool-centric. Networked ways to communicate, capture and manipulate information are essential to everyday work. At the same time the tools also affect the ways tasks are performed.

Furthermore, we argue that like in other development contexts, also in FLOSS development users and HCI experts should be able to enter the participative role, i.e. the role in which they are considered as equal partners in the design process with decision-making power regarding the solution [23, 35]. By relying on the model introduced by Hedberg and Iivari [33], Table 2 lists the recommended

Table 1. Summary of Communication Means Provided and Required

Tasks	**Support provided**	**Areas for Improvement**
Requirements construction (requirements, needs, problems asked for/provided)	Currently weak support. Feature requests may be provided in bug trackers, mailing lists, IRC or forums, but requirements are typically not constructed but asserted or only implied in the actual source code [56].	Tools supporting distributed requirements construction (gathering data on users, their needs, their contexts of use etc). 'Confusion reports' and 'surprise reports' [8], usability and user related discussion forums [12, 48, 61] suggested
Design (ideas, designs, implementations asked for/provided)	Currently weak support. Ideas may be provided in bug trackers, mailing lists, IRC or forums, but lack support for visualization or for collaboration	Tools supporting visualization and collaboration during distributed design work [30, 33]. E.g. user interface design by blogs [49, 60], open content projects [60], specific design areas supporting brainstorming [49, 60], a separate tab for usability/UX issues [5] suggested
Evaluation (feedback asked for/provided)	Currently some support. Bug trackers, mailing lists, IRC, forums etc. available for users to provide feedback, but may be difficult to use properly.	Tools supporting user and usability feedback gathering. E.g. moderators handling the user reported bugs [38], bug tracker guiding users in bug reporting [10], a tool for usability bug reporting [15, 49], reference/bleeding edge/core users providing feedback [60], open content projects [60], and remote usability testing [47–49] suggested

roles for human layer work in FLOSS projects, and connects these roles with the recommended communication tools (and practices), listing also numerous problems or needs they still have in connection to communication.

Even though there are communication means suggested for all the human layer roles, also problems and needs related to each role can be identified. For example, the UX-Tab for supporting the work of HCI core is explained in the article [5], but never implemented. Altogether, communication, coordination and decision

Table 2. Summary of communication means and problems connected to human layer roles

Role/Task	Tools (and practices)	Needs
HCI core	UX-tab [5]	An implementation of the UX-tab, a means to communicate with the technical core, ensuring decision-making power, tools supporting coordination related to UX and technical development
HCI designers	blogs [49, 60], brainstorming areas [49, 60]	Support for synchronous, collaborative, visual design work [33], tools supporting coordination related to UX and technical development
HCI evaluators	remote usability testing tools [47–49]	Tools supporting coordination related to UX and technical development
Technical users (additionally to non-technical user)	confusion reports [8], surprise reports [8], brainstorming areas [49, 60]	
Non-technical users	usability-forum [12, 48, 61], usability-tracker [15, 49], remote usability testing tools [47–49], open content projects [60], moderators [38]	More user-friendly tools [10]

making between HCI core and technical core need tool support and thus more investigating. Also there should be better support for coordination between the work of HCI designers and evaluators and technical development. The technical users in FLOSS projects are able to use the tools used by developers, but the non-technical users need more user-friendly solutions. By providing new tools for FLOSS development, the user-developer discussion could be facilitated, or even forced if it is built in a tool. However, it is not an easy task, because the FLOSS way of doing - the FLOSS philosophy - must be taken into account.

6 Conclusions

This paper examined the following questions: "What kinds of means for communication exist in FLOSS projects for user-developer communication? What kinds of means should there be?" The results indicate that user-developer communication during the evaluation task is somewhat supported in FLOSS projects,

but design and requirements construction tasks are badly in need for further support. Especially there is a need to support working of the HCI core team, as well as to support the coordination between HCI designers and evaluators and technical development.

The tools used in FLOSS development are not the best for managing user/UX related issues. Thus it seems that there is still room for tool development. However, we emphasize in line with Wilson that practices are in a bigger role than tools in software development [66]. Therefore, not only tool support is needed in FLOSS projects, but also new practices for user-developer communication. The tools can however enhance and stimulate to use those practices. Therefore, we have presented the status quo of user-developer communication tools and deductions on how to make the situation better by enhancing the tool set of FLOSS development.

The FLOSS philosophy must be taken into account as well. The developers should accept these new tools. Fogel presents two important points on the planning and taking in to use of new tools. First, it should not take too much effort. Secondly, users must think that the additional effort is worth it [27]. If this effort is too much, the new tool (or a feature of it) is omitted. It is necessary that new tools are not aimed to replace FLOSS practices, but to complement and make them better. Additionally they should be as transparent as the existing tools [21]. It seems that FLOSS developers value simple but versatile tools. Simple textual emails, wiki pages and bug tracker discussions do not dazzle with glorious visual effects, but they simply enable functions that are really needed.

The limitation of FLOSS research is that researchers tend to focus on large and well-known projects and communities. Small projects may not have same issues, but they should be studied also. Common trend is to make quantitative studies utilizing a data mass provided by FLOSS development support sites. For small projects qualitative case studies carried out might be better. Another limitation connected to this literature review is the varying terminology used in the studies addressing user-developer communication. It was very difficult to settle the appropriate keywords, since the studies have varyingly used terms such as user, usability, HCI, UX, usability engineering, user-centered design to describe their research focus. Even non-FLOSS projects are studied under FLOSS topic.

In addition to tool development, the paths for future work include looking for existing FLOSS projects. It would be helpful to extract the best practices, conventions and tools that have led to successful user-developer communication in FLOSS development from real cases. Even though there are some exploratory studies performed and enhancement ideas raised, it seems that the results are not used in practice in FLOSS development yet.

References

1. Andreasen, M., Nielsen, H., Schrøder, S., Stage, J.: Usability in open source software development: opinions and practice. Information Technology and Control 35(3A), 303–312 (2006)

2. Ankolekar, A., Herbsleb, J., Sycara, K.: Addressing Challenges to Open Source Collaboration With the Semantic Web. In: The 3rd Workshop on Open Source Software Engineering, the 25th International Conference on Software Engineering, ICSE, Portland OR, USA (2003)
3. Anvik, J., Hiew, L., Murphy, G.: Who Should Fix This Bug? In: Proceedings of the 28th International Conference on Software Engineering, ICSE 2006, pp. 361–370 (2006)
4. Bach, P.: Supporting the user experience in free/libre/open source software development, Ph.D. dissertation, Pennsylvania State University (2009)
5. Bach, P., DeLine, R., Carroll, J.: Designers Wanted: Participation and the User Experience in Open Source Software Development, Boston, MA, April 4-9, pp. 985–994. ACM, USA (2009)
6. Bach, P., Kirschner, B., Carroll, J.: Usability and Free/Libre/Open Source Software SIG: HCI Expertise and Design Rationale. ACM, New York (2007)
7. Bach, P., Twidale, M.: Lucky Seven: How Can the Crowd Help Design?, Penn State College of IST. University Park, PA (2007)
8. Bach, P., Twidale, M.: Involving reflective users in design. In: Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, April 10 - 15, ACM, New York (2040)
9. Benson, C., Müller-Prove, M., Mzourek, J.: Professional usability in open source projects: GNOME, OpenOffice.org, NetBeans. In: Extended Abstracts of CHI 2004, pp. 1083–1084. ACM, New York (2004)
10. Bettenburg, N., Just, S., Schrter, A., Weiss, C., Premraj, R., Zimmermann, T.: What makes a good bug report? In: Proceedings of the 16th ACM SIGSOFT international Symposium on Foundations of Software Engineering. ACM, New York (2008)
11. Beyer, H., Holtzblatt, K.: Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann Publishers, San Francisco (1998)
12. Bødker, M., Nielsen, L., Orngreen, R.N.: Enabling User Centered Design Processes in Open Source Communities. In: Aykin, N. (ed.) HCII 2007. LNCS, vol. 4559, pp. 10–18. Springer, Heidelberg (2007)
13. Bollinger, T., Nelson, R., Self, K., Turnbull, S.: Open Source Methods: Peering through the Clutter. IEEE Software 16(4), 8–11 (1999)
14. Carmel, E., Agarwal, R.: Tactical approaches for alleviating distance in global software development. IEEE Software 18(2), 22–29 (2001)
15. Cetin, G., Verzulli, D., Frings, S.: An analysis of involvement of HCI experts in distributed software development: Practical issues. In: Schuler, D. (ed.) HCII 2007 and OCSC 2007. LNCS, vol. 4564, pp. 32–40. Springer, Heidelberg (2007)
16. Cheung, G., Chilana, P., Kane, S., Pellett, B.: Designing for discovery: opening the hood for open-source end user tinkering. In: Proceedings of the 27th international Conference Extended Abstracts on Human Factors in Computing Systems, CHI 2009, pp. 4321–4326. ACM, New York (2009)
17. Cooper, C., Bowers, J.: Representing the users: notes on the disciplinary rhetoric of human-computer interaction. In: Thomas, P.J. (ed.) The Social and Interactional Dimension of Human-Computer Interfaces, pp. 48–66. Cambridge University Press, Cambridge (1995)
18. Cooper, A., Reimann, R.: About Face 2.0: The essentials of interaction design. In: Information Visualization, vol. 3, pp. 223–225. New Wiley Pub., Indianapolis (2004)

19. Crowston, K., Annabi, H., Howison, J., Masango, C.: Effective work practices for floss development: A model and propositions. In: Proceedings of the 38th Hawaii International Conference On System Sciences, HICSS 2005, IEEE Press, Piscataway (2005)
20. Crowston, K., Howison, J.: The social structure of free and open source software development (2005), http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/viewArticle/1207/1127 (retrieved on September 1, 2009)
21. Čubranić, D., Booth, K.S.: Coordinating Open-Source Software Development. In: Proceedings of the 8th IEEE International Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE 1999, pp. 61–65. IEEE CS Press, Los Alamitos (1999)
22. Curtis, B., Krasner, H., Iscoe, N.: A Field Study of the Software Design Process for Large Systems. Communications of the ACM 31(11), 1268–1287 (1988)
23. Damodaran, L.: User involvement in the systems design process - a practical guide for users. Behaviour & Information Technology 15(16), 363–377 (1996)
24. Fischer, G.: Communities of Interest: Learning through the Interaction of Multiple Knowledge Systems. In: 24th Annual Information Systems Research Seminar In Scandinavia, IRIS 24, Ulvik, pp. 1–14 (2001)
25. Fitzgerald, B., Ågerfalk, P.: The Mysteries of Open Source Software: Black and White and Red All Over? In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS 2005. IEEE Press, Piscataway (2005)
26. Fogel, K.: Producing Open Source Software. O'Reilly, Sebastopol (2005), http://producingoss.com/ (retrieved on December 15, 2009)
27. Fogel, K.: Beautiful Teams: Inspiring and Cautionary Tales from Veteran Team Leaders. ch. 21. O'Reilly, Sebastopol (2009), http://www.red-bean.com/kfogel/beautiful-teams/bt-chapter-21.html (retrieved on September 1, 2009)
28. Free Software Foundation, Inc., The Free Software Definition (2008), http://www.gnu.org/philosophy/free-sw.html (retrieved on: September 1, 2009)
29. Ge, X., Dong, Y., Huang, K.: Shared Knowledge Construction in an Open-Source Software Development Community: An Investigation of the Gallery Community. In: Proceedings of the International Conference on Learning Sciences, Bloomington, IN, June27-July 1, pp. 189–195 (2006)
30. Geisler, C., Rogers, E.: Technological Mediation for Design Collaboration. In: Proceedings of the IEEE Professional Communication Society International Professional Communication Conference and Proceedings of the 18th Annual ACM International Conference on Computer Documentation: Technology & Teamwork, IPCC/SIGDOC 2000. IEEE Educational Activities Department, Piscataway (2000)
31. Halloran, T.J., Scherlis, W.L.: High Quality and Open Source Practices. Presented at the 2nd Workshop on Open Source Software Engineering, Orlando, FL (2002)
32. Hartson, H.R., Castillo, J.C., Kelso, J., Neale, W.C.: Remote evaluation: the network as an extension of the usability laboratory. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Common Ground, CHI 1996, pp. 228–235. ACM, New York (1996)

33. Hedberg, H., Iivari, N.: Integrating HCI Specialists into Open Source Software Development Projects. In: Boldyreff, C., Crowston, K., Lundell, B., Wasserman, A.I. (eds.) OSS 2009. IFIP AICT, vol. 299, pp. 251–263. Springer, Heidelberg (2009)
34. Howison, J.: Studying Free Software with Free Software and Free methods. In: A paper for the Australian Open Source Development Conference, Melbourne, Australia (December 1-3, 2004)
35. Iivari, N.: Representing the User' in software development - a cultural analysis of usability work in the product development context. Interact. Comput. 18(4), 635–664 (2006)
36. Iivari, N.: Constructing the users' in open source software development: An interpretive case study of user participation. Information Technology & People 22(2), 132–156 (2009)
37. Iivari, N., Molin-Juustila, T.: Listening to the Voices of the Users, in Product Based Software Development. International Journal of Technology and Human Interaction 5(3), 54–77 (2009)
38. Ko, A., Chilana, P.: How Power Users Help and Hinder Open Bug Reporting, In. In: Proceeding of the 28th International Conference on Human Factors in Computing Systems, CHI 2010. ACM, New York (2010)
39. Kittur, A., Kraut, R.E.: Harnessing the wisdom of crowds in wikipedia: quality through coordination. In: Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work, CSCW 2008, ACM, New York (2008)
40. Lakhani, K., von Hippel, E.: How open source software works: 'free' user-to-user assistance. Research Policy 32, 923–943 (2003)
41. Lievrouw, L.: Oppositional and activist new media: remediation, reconfiguration, participation. In: Proceedings of the Participatory Design Conference, pp. 115–124. CPSR, Palo Alto (2006)
42. Ljungberg, J.: Open Source Movements as a Model of Organising. In: Proceedings of 8th European Conference on Information Systems, Vienna, pp. 208–216 (2000)
43. Massey, B.: Where Do Open Source Requirements Come From (And What Should We Do About It)? In: Proceedings of the 2nd Workshop on Open Source Software Engineering, ICSE (2001)
44. Mayhew, D.: The usability engineering lifecycle: a practitioner's handbook for user interface design. Morgan Kaufmann Publishers Inc., San Francisco (1999)
45. Mockus, A., Fielding, R., Herbsleb, J.: A Case Study of Open Source Software Development: The Apache Server. In: Proceedings of the 22nd International Conference on Software Engineering, ICSE 2000. ACM, New York (2000)
46. McConnell, S.: Open source methodology: ready for prime time? IEEE Software 16(4), 6–8 (1999)
47. Nichols, D., McKay, D., Twidale, M.: Participatory Usability: supporting proactive users. In: Proceedings of the 4th Annual Conference of the ACM Special Interest Group on Computer Human Interaction, pp. 63–68. ACM, Dunedin (2003)
48. Nichols, D., Twidale, M.: The Usability of Open Source Software. First Monday 8(1) (2003)
49. Nichols, D., Twidale, M.: Usability processes in open source projects. Software Process Improvement and Practice 11, 149–162 (2006)
50. Ogawa, M., Ma, K., Bird, C., Devanbu, P., Gourley, A.: Visualizing Social Interaction in Open Source Software Projects. In: Proceedings of Asia-Pacific Symposium on Visualization, APVIS, pp. 25–32 (February 2007)

51. Open Source Initiative, The Open Source Definition (2006), http://www.opensource.org/docs/osd (retrieved on September 1, 2009)
52. Parkinson, C.N.: Parkinson's law: or, the pursuit of progress / C. Northcote Parkinson J. Murray, London (1957)
53. Paul, C.: A Survey of Usability Practices in Free/Libre/Open Source Software. In: Boldyreff, C., Crowston, K., Lundell, B., Wasserman, A.I. (eds.) OSS 2009. IFIP AICT, vol. 299, pp. 264–273. Springer, Heidelberg (2009)
54. Raymond, E.: The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. OReilly, US (1999)
55. Rosson, M., Carrol, J.: Usability Engineering: Scenario-Based Development of Human-Computer Interaction. Morgan Kaufmann, New York (2002)
56. Scacchi, W.: Understanding the requirements for developing open source software systems. IEE Proceedings - Software 149(1), 24–39 (2002)
57. Schwartz, D., Gunn, A.: Integrating user experience into free/libre open source software: CHI 2009 special interest group. In: Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems, CHI 2009, pp. 2739–2742. ACM, New York (2009)
58. Singh, V., Twidale, M.B., Nichols, D.M.: Users of Open Source Software - How Do They Get Help? In: Proceedings of the 42nd Hawaii International Conference on System Sciences, HICSS 2009, Big Island, Hawaii, January 5-8 (2009)
59. Stamelos, I., Angelis, L., Oikonomou, A., Bleris, G.: Code quality analysis in open source software development. Information Systems Journal 12, 43–60 (2002)
60. Terry, M., Kay, M., Lafreniere, B.: Perceptions and Practices of Usability in the Free/Open Source Software (FOSS) Community. In: Proceedings of the Conference on Human Factors in Computing Systems, pp. 999–1008. ACM, New York (2010)
61. Twidale, M., Nichols, D.: Exploring usability discussions in open source development. In: Proceedings of the 38th Hawaii International Conference on System Sciences, HICSS 2005. IEEE Press, Piscataway (2005)
62. Viorres, N., Xenofon, P., Stavrakis, M., Vlachogiannis, E., Koutsabasis, P., Darzentas, J.: Major HCI challenges for open source software adoption and development. In: Schuler, D. (ed.) HCII 2007 and OCSC 2007. LNCS, vol. 4564, pp. 455–464. Springer, Heidelberg (2007)
63. von Hippel, E., von Krogh, G.: Open source software and the "private-collective" innovation model, Issues for organization science. Organ. Sci. 14(2), 209–223 (2003)
64. Wiggins, A., Howison, J., Crowston, K.: Social dynamics of FLOSS team communication across channels. In: Proceedings of the IFIP 2.13 Working Conference on Open Source Software (OSS), Milan, Italy, pp. 131–142 (2008)
65. Williams, S.: Free as in Freedom: Richard Stallman and the Free: Richard Stallmans Crusade for Free Software. O'Reilly Media, Sebastopol (2002); ISBN: 0596002874
66. Wilson, G.: Is the Open-Source Community Setting a Bad Example? IEEE Software 16(1), 23–25 (1999)
67. Yamauchi, Y., Yokozawa, M., Shinohara, T., Ishida, T.: Collaboration with Lean Media: how open-source software succeeds. In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, CSCW 2000, pp. 329–338. ACM, New York (2000)
68. Yatani, K., Chung, E., Jensen, C., Truong, K.N.: Understanding how and why open source contributors use diagrams in the development of Ubuntu. In: Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, pp. 995–1004. ACM, New York (2009)

69. Ye, Y., Kishida, K.: Toward an understanding of the motivation of open source software developers. In: Proceedings of the 25th International Conference on Software Engineering (ICSE), pp. 419–422. IEEE Press, Piscataway (2003)
70. Zhao, L., Deek, F.: Improving open source software usability. In: Proceedings of the 11th Americas Conference on Information Systems, AMCIS, Omaha, NE, August 11-14, pp. 923–928 (2005)