Perceptions and Practices of Usability in the Free/Open Source Software (FOSS) Community

Michael Terry, Matthew Kay, Ben Lafreniere
HCI Lab, David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada
mterry@cs.uwaterloo.ca

ABSTRACT
This paper presents results from a study examining perceptions and practices of usability in the free/open source software (FOSS) community. 27 individuals associated with 11 different FOSS projects were interviewed to understand how they think about, act on, and are motivated to address usability issues. Our results indicate that FOSS project members possess rather sophisticated notions of software usability, which collectively mirror definitions commonly found in HCI textbooks. Our study also uncovered a wide range of practices that ultimately work to improve software usability. Importantly, these activities are typically based on close, direct interpersonal relationships between developers and their *core users*, a group of users who closely follow the project and provide high quality, respected feedback. These relationships, along with positive feedback from other users, generate *social rewards* that serve as the primary motivations for attending to usability issues on a day-to-day basis. These findings suggest a need to reconceptualize HCI methods to better fit this culture of practice and its corresponding value system.

Author Keywords
Reference users, bleeding edge users, core users

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Design

INTRODUCTION
In the past 10 years, free/open source software (FOSS) has grown to be a vital component of the computing landscape: It increasingly forms the cornerstone of IT infrastructure in business, education, and government [23]; it powers commercial products such as TiVo, Mac OS X, and netbooks; and it has created a multi-billion dollar service industry for companies like Red Hat, Novell, and IBM [22].

CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.

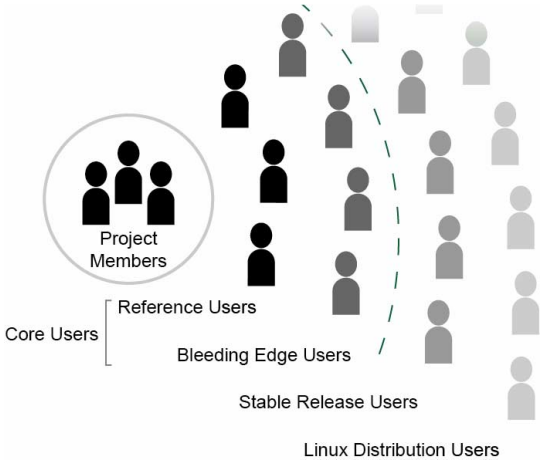

Figure 1. Strata of users for an open source project. Reference users and bleeding edge users comprise the project's "core users" and play pivotal roles in FOSS usability processes.

In recent years, the FOSS community has turned its attention to improving its usability practices [5,7,14,16,20,27]. Noting this trend, past research has identified some of the challenges inherent in addressing usability in FOSS development. For example, the distributed, largely voluntary nature of FOSS development makes it difficult to engage in holistic design methods since developers tend to work on individual application components in isolation [3,5,16].

Despite these challenges, the FOSS community is gradually adopting usability techniques that mesh with their development practices. For example, Twidale and Nichols found that some projects now engage in a practice they dub "design-by-blog" whereby prospective designs are posted on personal blogs to solicit feedback from the software's user base [17].

Past work provides an important foundation for understanding the challenges and practices associated with addressing usability issues in FOSS development. However, much of this research has been conducted at a distance, often by examining bug reports and mailing list archives. As such, there is little information about how members of this community actually perceive usability issues, or why they may be motivated to address these issues in software development. Furthermore, past research has examined only a

handful of projects. As open software development becomes an increasingly common way of producing software, both the FOSS and HCI communities would benefit from a deeper understanding of how FOSS project members think about, act on, and are motivated by usability issues.

This paper presents results from a study investigating perceptions and practices of usability in the FOSS community. In total, 27 individuals associated with 11 different FOSS projects were interviewed to understand how they conceptualize the notion of usability, how it is addressed in projects, and the motivations for doing so. Participants included project members (developers, user experience (UX) engineers, documentation writers, and localization contributors) and end-users closely connected to the project, a group we call the project's *core users*. As we will show, core users play a crucial role in FOSS usability.

Across the projects and participants, a number of common themes arose.

First, counter to common perception, we found project members possess rather sophisticated notions of the concept of usability. In fact, taken collectively, their definitions of usability closely match those of typical HCI textbooks.

Second, our interviews reveal a range of both ad-hoc and structured practices that ultimately serve to improve the software's overall usability. While previous work has noted a handful of emerging practices in this community, this research provides a much more extensive view of current FOSS usability practices, along with their salient features. The most important of these practices are based on the rich interactions between project members and their core users. Core users include *reference users*, or users who are valued for their domain expertise and their experience in using the software; and *bleeding edge users*, or users who track and use nightly builds of the software (Figure 1). Both user groups provide high quality, respected feedback, which is often explicitly solicited by developers as they design and implement new functionality. The relationships that form between the developers and their core users ultimately contribute to an organic form of participatory design driven by the real-time needs of both parties.

Finally, our results indicate that in the absence of economic incentives, FOSS developers are motivated to address usability concerns by the high quality, positive feedback they receive from their core users and other trusted users. This finding suggests one cannot assume that traditional motivations for HCI are sufficient, or even relevant, to compel FOSS projects to address usability concerns. For example, it is often assumed that the possibility of increasing the software's user base is sufficient to motivate FOSS developers to address usability issues (e.g., [1,16]). In fact, we found that a large user base can actually act as a disincentive since it increases the number of bug reports, feature requests, and complaints. Instead, our study suggests that the more important currency for motivating usability work in the FOSS community is social rewards – praise and positive feedback from end-users whose opinions are valued.

Collectively, these results argue for the need to transform HCI methods to better match this culture of practice and its corresponding value system, as has been suggested in the past [2,16]. Importantly, our work identifies one of the key ways in which HCI methods need to change: In the absence of economic incentives, HCI methods need to be reformulated to foster and take advantage of the social relationships that arise between FOSS developers and their end-users, because it is these relationships that motivate attention to usability issues in day-to-day development. Reframing HCI as an activity that occurs within an open development environment, in which developers and end-users have direct, ongoing social relationships, reveals a large, unexplored space for new HCI methods.

The rest of the paper examines these issues in greater detail, beginning with a survey of related work, followed by a description of the study and its methods. We then relay the results of the study by describing how respondents define usability, how they discover and address usability issues, and their motivations for doing so. We conclude with implications for improving usability practices in the FOSS community.

BACKGROUND

In this section, we provide a general overview of FOSS development practices, then describe past work examining usability issues in the FOSS community.

FOSS Development

FOSS is typically developed in a highly distributed fashion, with tools such as mailing lists, IRC, source code repositories, and bug reporting systems used to support communication and synchronization of work practices [8,13,23]. Text is the primary medium of communication as well as the primary object of interest (more specifically, source code).

Project membership and organizational structures in FOSS projects are typically based on merit [6,7,21], and contributors are often volunteers who choose tasks based on personal interests [13,29]. The volunteer nature of the community has led to considerable research examining the motivations for contributing to FOSS projects. Among other reasons, past work has found that project members contribute to develop skills [19,24,29]; for the intellectual challenge of the activity [23]; to build social capital and improve one's reputation [9,19,24]; and to be part of a community [18].

The underlying software architecture of FOSS projects tends to be quite modular in nature, with developers overseeing all modifications to the modules they "own" [13,23]. This modularity helps in attracting new developers, since it is easier to start making contributions to a small, independent portion of code than to a large, monolithic application [4].

Usability in the FOSS Community

As the free/open source software community has grown its user base beyond software developers, it has become increasingly interested in creating well-designed, usable software [5,7,16].

Early contributions to FOSS usability efforts were driven by corporations such as Sun and Novell, who provided human interface guidelines (HIGs) [5] and results of usability tests [7]. The resultant HIGs have proved to be extremely useful, as they serve as an authoritative reference for some matters of interface design [5,17].

While corporate contributions have played an important role in jump-starting usability efforts in the FOSS community, a number of challenges have been identified for more fully incorporating usability methods and expertise in individual projects. These challenges arise due to the way in which FOSS is developed, as well as its developer-centric culture, as we describe.

Free/open source software architectures are often highly modular in design. However, usability concerns cut across the entire application. As a consequence, it can be difficult for individuals to make small, incremental improvements to a software's usability, a significant issue for software largely developed by volunteers in their spare time [3,5,16]. Similarly, the lack of dedicated infrastructure for usability activities and design artifacts makes it difficult to coordinate usability work in distributed development environments [3,16]. Finally, since this merit-based culture has traditionally valued source code as its primary currency, it has sometimes been difficult for UX practitioners to join and make contributions that are perceived as valuable [3,16].

Despite these challenges, the community is gradually developing techniques to better address usability concerns. One technique already mentioned is "design-by-blog," in which informal design critiques are held on personal blogs [17]. Projects have also been found to use screenshots, annotated screenshots, and ASCII art to support design discussions by email and in bug trackers [28]. Finally, UX experts are slowly being integrated into projects, and are partially responsible for the emergent usability practices, such as design-by-blog and the creation of dedicated usability infrastructure (e.g., wikis, mailing lists) [3].

This prior research reveals the unique features and practices of the FOSS community. However, past work examining FOSS usability has considered only a handful of projects, and has typically used only mailing list archives and bug reports as its source material. To better understand how the community thinks about and acts on usability, we interviewed members of the FOSS community about these issues. We describe this study next.

STUDY DESCRIPTION

In this section, we describe the primary research questions of the study, the study methods, and the study participants.

Research Questions

In this research, we were interested in examining the following research questions:

- How do open source developers define and conceptualize the notion of usability?
- What motivations do FOSS developers have for creating software that is usable by people other than themselves?
- What are current usability practices in the FOSS community?
- How do FOSS usability practices differ from traditional usability practices?

Throughout this paper, we use the term "usability" as a generic term to encompass any and all human-centric concerns in software design. We took a similar approach in our interviews, as we describe below.

Methods

Three researchers conducted interviews at a FOSS conference, at a corporation that produces FOSS software, and remotely via Skype.

Subjects were recruited at a FOSS conference and by directly contacting individual FOSS projects. Our primary goal in recruiting was to interview a representative sample of project members (developers, documentation writers, localization contributors, etc.) across a range of projects that varied in terms of the type of software produced, the maturity of the project, and their organizational structure. Since software usability necessarily involves users, we also wished to understand FOSS users' perspectives on these matters. Thus, we interviewed users who interact regularly with project members to understand their potential role in influencing the usability of the FOSS applications they use, as well as their thoughts on these issues.

Interviews typically lasted one hour. All interviews were recorded, except for one interview in which the participant did not provide consent for use of a recording device (notes were taken by hand instead).

Each interview was divided into the following segments:

1. Obtaining basic background information on the participant, such as their day job and what FOSS project they are associated with
2. Learning how and when they got involved with their FOSS project. If they were a project member, we also asked what they do in the project and why they stay involved
3. Their perception of the concept of "usability"

Project Code	Project description	Project organization	Participants interviewed
3DA	A 3D animation package	Largely volunteer-driven, with some project support provided by an associated non-profit corporation	D9, NCC3, CU1
BG	A bitmap graphics application	Volunteer-driven	D2, D3, D4, D8, UX1, NCC1, NCC4
CMS	A content management system for the web	Largely volunteer-driven, with some project support provided by an associated non-profit corporation	UX4
DTP	Desktop publishing application	Volunteer-driven	D1, D5, NCC2, CU4
DUT	A desktop utility	Volunteer-driven	D10
DWE	A desktop windowing environment	Volunteer-driven	UX5
FE	A font editor	Volunteer-driven	CU2, CU3
OS	A desktop operating system	Overseen by a commercial company, though the majority of contributions are estimated to come from volunteers	D10, D11, UX3
VE	A video editing application	Was volunteer-driven, but now includes paid developers	NCC5
VG	A vector-based graphics application	Volunteer-driven	D7, CU5
WB	A web browser	Includes paid project members and a large volunteer developer base. Overseen by a for-profit corporation	D6, D12, UX2

Table 1: Projects and project codes

4. How they practice or perceive others practicing usability in the project

We used the term "usability" throughout the interview without defining it. We chose the word because we felt it was suitably generic that each participant could define it in any way they wished. In fact, some objected to the term and offered terms they felt were superior (such as "user experience").

All three researchers analyzed the first ten interviews to identify common themes. The primary researcher then analyzed all interviews. For each interview, a document was created summarizing the participant's responses to the segments of the interview. Individual quotes were extracted from these summaries then clustered to identify common themes using a bottom-up, inductive analysis approach. Throughout the process, the emergent themes were regularly validated by all three researchers in dedicated interpretation sessions.

Participants

We interviewed 12 developers, 5 user experience (UX) project members, 5 non-code contributing project members (documentation writers, localization contributors), and 5 core users. All individuals were members of their respective software project's project team, with the exception of the core users.

As mentioned in the Introduction, we use the phrase "core users" to refer to motivated users who closely follow and interact with the project. This group of users includes *reference users*, or users with close social ties to individual project members, with domain expertise and extensive experience using the software, and *bleeding edge users*, or users who use nightly builds of the software. While past research has identified numerous roles in FOSS development (such as core members, peripheral developers, and bug reporters [15]) and has also distinguished between active and passive users (e.g., those who submit bug reports and feature requests, and those who don't) [10], we are unaware of prior work that distinguishes users in these ways. These classification schemes arose from our data and, as we will show, describe groups of users who play a vital role in FOSS usability practices.

Throughout the paper, we use the codes "D#", "UX#", "NCC#", and "CU#" (where # represents a unique number) to refer to developers, user experience engineers, non-code contributing project members, and core users, respectively. We also use codes for each project.

A quick note is in order with respect to the presentation of research results and our use of project codes in this paper. FOSS projects are typically conducted "in the open" on the Internet, with communications stored in easily searched, publicly accessible archives. This openness makes it difficult to completely anonymize project descriptions. However, the contributions of this work lie in characterizing trends and patterns within the FOSS community, rather than in attributing specific behaviors to specific projects. Thus, while we recognize that a determined individual may be

Table 2: Definitions of Usability

(12) **Learnability/discoverability**. Features can be learned, discovered (D1, D7, D9, D10, NCC2, NCC3, NCC4, NCC5, UX5, CU1, CU3)

(10) **Logical**: Interface and interaction follow a certain logic, a set of rules, and things work as expected (D1, D2, D6, D8, NCC3, NCC4, UX2, CU1, CU2, CU3)

(8) **Efficiency**. Users can perform tasks efficiently. Often described in context of high-end and/or professional users (D1, D3, D9, D10, NCC3, NCC4, CU1, CU4)

(7) **Simplicity**. Interface design and presentation is simple. This term mostly refers to a design aesthetic rather than the application's feature set (D4, D7, D9, D10, D11, NCC3, NCC5)

(6) **Transparency**. The tool doesn't get in the way of the user, and fades into the background (D4, D8, D10, D12, UX2, UX3)

(6) **Consistency**. The design of interface and its behavior are consistent (D1, D4, D11, NCC3, CU1, CU3)

(5) **Intuitive**. How one performs tasks matches one's intuition (D9, D11, NCC2, CU1, CU3)

(4) **Easy-to-use**. Software is easy-to-use (D3, D5, D9, CU3)

(3) **Low floor, high ceiling**. Simple things are easy, complex things possible (D7, D8, CU1)

(2) **Meshes with workflow** (NCC3)

(2) The software is **usable**. (D7, CU2)

(2) **Minimal cognitive load**. Interface minimizes stress and cognitive load on the user (D10, D12)

(2) **Understanding user needs**. Usability refers to "getting in the user's head space," understanding their wants and expectations; understanding the target user group; and user-centered design (UX2, UX5)

(2) **Holistic quality**. The interface, interaction, and visual design form a holistic unit (UX3, UX4)

(1) **Memorability**. Users can remember how to perform a task (UX5)

(1) **Easily accessible functionality**, meaning functionality is visible and how to use it is obvious to end-user (CU5)

(1) **Real-world metaphors**. The interface makes use of real-world metaphors (NCC2)

(1) **Less support**. A usable interface results in fewer questions on IRC or mailing lists (D1)

(1) **Enables creativity** (CU4)

(1) **Term is inadequate**; "user experience" is the more appropriate concept to consider (UX4)

able to surmise the true identity of some projects studied, we still refer to each via a code.

Table 1 briefly summarizes each project, the basic type of organizational structure, its project code, and the participants associated with the project. With this in place, we now turn to the study results.

DEFINITIONS OF USABILITY

Table 2 presents participants' definitions of usability, ordered according to the number of respondents mentioning a particular concept. Participants were free to define the concept in any way they pleased, so some participants cited multiple concepts in their definitions. The number of respondents for each concept is listed first in parentheses, followed by the concept and a list of participants mentioning that concept.

As can be seen, the definitions of usability span the range of definitions commonly found in the HCI textbooks, and demonstrate that the community, as a whole, possesses a fairly sophisticated, well-rounded notion of the concept. (By way of comparison, Shneiderman *et al.*'s introductory HCI textbook offers learnability, efficiency, error rates, memorability, and subjective satisfaction as usability measures [25], most of which are either explicitly referenced or alluded to by our participants.)

Learnability/discoverability was most frequently offered as a component of usability, though many participants noted that for high-end software, being learnable should not come at the cost of power. This desire for power in FOSS applications has been previously noted [16], but in our study, participants cited a concomitant need to provide power that can be *efficiently* used.

A particularly common theme among the participants was the notion that a usable interface exhibits some form of logic in its design and use. This sentiment was echoed by participants who mentioned "consistency," for a total of 12 different respondents. This emphasis on a usable design possessing an inherent logic was unexpected as it typically receives only minor treatment in HCI texts (for example, Shneiderman *et al.* devote only one page to "consistency" in their text [25]). However, this emphasis may arise due to a similar desire for clean, elegant, and logical code [12].

Usability and User Needs

Across the participants, only two explicitly introduced the notion of user needs in their definitions of usability (UX2, UX5). Both of these individuals have an educational background in human-centered computing. However, this observation should not be interpreted as other project members being ambivalent to their software's usability (there is ample evidence that they are very concerned about its usability). Instead, there appears to be less of a perceived need to *actively discover* user needs. There are a number of potential explanations for this finding that can be drawn from our interview data.

First, for mature projects, such as 3DA, BG, and VG, their software already offers sophisticated functionality. For them, the challenge is often providing this functionality in a way that seamlessly integrates with users' workflows.

Second, project members engage in an ongoing dialogue with their core users about the software, its functionality,

how it is used, and how it should be improved. Similarly, projects continually receive bug reports and feature requests in mailing lists, IRC, and bug tracking databases. These continual streams of user input likely lessen the perceived need to *actively* discover user needs, especially since the existing bugs and feature requests often outstrip a project's resources.

Finally, the limited attention paid to discovering user needs may also be due to a lack of market pressures, coupled with the volunteer nature of the projects. That is, projects are not driven to compete in the marketplace in the same way commercial products are. We examine this issue in greater detail later when discussing motivations for usability.

From these conceptualizations of usability, we now turn to the topic of usability practices.

USABILITY PRACTICES: DEVELOPER-USER RELATIONSHIPS AND THE ROLES OF UX MEMBERS

With the exception of the dedicated UX project members, few project members claimed to do much to discover or address usability issues. However, we consistently found project members report activities that directly or indirectly contribute to the usability of the software. Table 3 summarizes the methods we identified that contribute to software usability.

As can be seen in the table, a wide range of practices were uncovered, including those common to commercial software development (such as the use of HIGs). Most notably, a number of practices are the result of the highly accessible nature of the project members. In particular, direct interactions between developers and end-users lead to a number of practices similar to participatory design. In this section, we focus primarily on these direct interactions between developers and end-users, and discuss how project members communicate with end-users about usability issues; how they rely on reference users and bleeding edge users to test developing versions of the software; and how Open Content Projects (projects aimed at producing content using the software) pair developers and core users on joint projects. Finally, we consider current roles of UX members in FOSS projects.

Communicating with Users

By far, the most commonly cited means of discovering and discussing usability issues was through the project's communication channels, specifically, IRC and mailing lists. In these contexts, users engage in an ongoing, direct dialogue with the developers about perceived issues and needs. IRC is especially useful in this respect because of its real-time nature, but mailing lists provide similar benefits. This form of direct interaction between developers and users is natural in the open development environment of the FOSS community, and is an example of a practice without a strong analogue in closed source development. Instead, in closed source contexts, these types of direct interactions must be explicitly staged.

Getting to the Heart of Usability Issues

In discussing usability issues with their users, project members often cited the need to dig deeper to understand the user's true task. NCC2 describes this process of probing user needs, and expresses the frustration that can come in trying to help users:

> *Usually, the hardest part is teasing out, from the person who is complaining, something about what they're complaining about, that you can understand. You get a lot of venting about something. You get people telling you that, "This program sucks. I've tried to use DTP, and I've used InDesign, and InDesign is so much better…" But that's not helpful information. You'd like to say, "Tell us what you're trying to do"… It's hard to get them beyond "Well, I want to do this, and I want the computer when I do this to do that…" It's like, Well, let's step back a little bit and say, "What is it you're trying to accomplish in your layout to go from this point to that point?" Forget about the operations – they're focused on the operations that they've commonly used before – and it's hard to get that out of people. Usually it takes a lot of work, and I would say most of the time you fail trying to get something you can understand, and if you can't understand it, it's hard to address it.*

This quote was representative of a number of project members' perspectives, and demonstrates an earnest desire to help users. It also demonstrates recognition of the importance of clarifying users' true tasks. Once those needs are understood, projects often design to those needs in a way that matches the overall interaction design of the application, rather than blindly cloning a commercial application. For example, D7 relayed a story about how a request for a feature found in Adobe Illustrator was initially rejected by one project member, but eventually implemented (in a form suitable for VG's design) after D7 talked with the user on IRC to understand his task. This exchange and the initial implementation of the new functionality occurred over the course of a few days, and is illustrative of how end-users directly engage developers in FOSS projects to negotiate the addition or modification of functionality for their own real-time needs.

Reference Users, Bleeding Edge Users, and Graduated Testing

A common perception of the FOSS community is that developers build software for themselves, to "scratch their own itch" (e.g., [21,27]). However, in our study a number of developers reported that they do not regularly use the software they produce, and often lack relevant domain expertise. D8 illustrates this finding:

> *One of the problems is that we don't really use our own program. We're working on the program because it's an interesting challenge to design an image manipulation application, but we don't usually do a lot of graphics. This is a problem because we just implement something, test it, then maybe never use it again. There is this detach between people using BG daily and us.*

Methods for Discovering Usability Issues	Methods for Addressing Usability Issues
• By paying attention to what is asked, discussed, or requested in internet relay chat (IRC), mailing lists, and forums *(all projects)* • Through bug reports *(all projects)* • By discovering what doesn't work for them as they develop the software *(all projects)* • Via "reference users", "bleeding edge" users of nightly builds, and professional users *(3DA, BG, DTP, DTU, FE, VG, WB)* • Through informal observations of friends and family; observations of conference attendees using software at project booths; or watching talks at conferences where high-end users give demos of workflows *(BG, DTU, VG, WB)* • By finding inconsistencies in the "rules" of the interface or one's expectations *(3DA, BG, DTU)* • By conducting think-aloud studies *(DTU, OS, WB)* • By performing usability studies in controlled settings *(CMS, DWE, OS)* • By writing a manual, documentation, or a book and relaying discovered issues, or by viewing how long it takes to explain concepts in documentation written by others *(3DA, DTP)* • By giving tutorials on the software or getting reports of difficulties others encountered when teaching the software in classes *(BG, DTP)* • Via expert reviews, performed remotely by UX members *(DWE, VE)* • Through Open Content projects *(3DA)* • Interviews of users *(DWE)* • Through surveys of user base *(WB)* • Directed program asking for users to identify small usability problems that can be quickly fixed *(OS)*	• Discussions on IRC and mailing lists *(all projects)* • Seeking feedback on mock-ups, prototypes, and custom builds from others (including reference users and bleeding edge users) *(3DA, BG, CMS, DTP, DTU, OS, VE, WB)* • Drawing up specifications, setting milestones, or articulating visions *(3DA, BG, OS, VG, WB)* • Via dedicated UX people *(BG, CMS, DWE, OS, WB)* • Use of Human Interface Guidelines (HIGs) *(BG, DWE, OS, VE)* • Annual, monthly, weekly, or ad hoc meetings, either in person or on IRC *(3DA, BG, DWE, OS)* • Defining a target user group *(3DA, BG, DTP, VG)* • Through usability "champions" in the project *(3DA, VG)* • Reliance on the larger community of users to fill in gaps in expertise and/or lack of equipment *(3DA, DTP)* • Creating personas of users *(OS, WB)* • Blogging about designs, getting feedback from user base *(OS, WB)* • Participation in the Season of Usability *(DWE, OS)* • By fixing things with solutions that are "logically" better *(VG)* • Creating scenarios of use to guide design *(BG)* • Running design clinics at conferences *(OS)* • Open Content Projects *(3DA)* • By developing and applying UI design patterns *(DWE)* • Interface brainstorming wiki *(BG)*

Table 3: Methods for Discovering and Addressing Usability Concerns

For these reasons and others, the project members often rely on core users to engage in a process we call *graduated testing*. In this process, new designs are incrementally evaluated by increasingly larger groups of users. First, reference users and bleeding edge users test development versions of the software. Then, a larger group of users use the software and provide feedback when it is released as a stable version. Finally, the last significant group of users is reached when the software is included in Linux distributions (Figure 1). While this last user group (those who only use software provided in their Linux distribution) may not be the largest, it is the last set of users to receive the latest version of software. However, by the time stable release users and Linux distribution users receive the software, it has undergone many rounds of tests.

D8 describes the early stages of this testing process:

The first thing is, of course, testing it yourself, because you're just, right now, playing with the code, and of course, you want to figure out, does it feel right for you, because if it doesn't feel right for you, it doesn't make sense to put it on others. Then I would check it in, and I had some people who were kind of our reference users, who we could ask, "What do you think about this?" …They might give feedback [like], I don't care if it's this way or that way, but also [feedback on how to improve the design]. [These reference users] were two people who are also connected with BG for quite a long time. We met each other frequently at BG conferences, so there was a certain trust. They also had a reputation for creating graphics for the GNOME project, and obviously for the BG user interface… So basically, they were very close to the project, so "reference artists" was kind of the internal designation.

This process of explicitly soliciting feedback from users close to the project also came up in discussions with members of the 3DA, DTP, DTU, FE, VG, and WB projects. As such, this practice constitutes one of the more important methods we found for discovering and addressing usability issues in FOSS development. It also illustrates another example of the ways in which developers and end-users di-

rectly interact with one another through software development.

One of the chief advantages of working closely with reference users and bleeding edge users is that developers obtain high quality, *targeted* feedback from a relatively small set of trusted, knowledgeable users early on, before the features become part of the official, stable version of the software. Accordingly, major usability issues can be detected and corrected before the software reaches a large audience. These pre-existing relationships also lessen the need to establish shared context or goals, increasing the quality of the communications.

One side-effect of this practice is that it can potentially lead to software tuned to the needs of users close to the project, rather than the larger user base, an issue that has been noted in the past [5]. This possibility echoes findings of Mockus *et al.*, who found that the Apache project pays the most attention to problems reported on the mailing list since those reporting issues are likely to be part of the developer community and thus able to provide "sufficient information to analyze the problem" [13].

Open Content Projects

While we found many projects engage in similar, ad-hoc practices, the 3DA project had one highly effective, unique approach for joining developers and users together to improve the software. Specifically, the 3DA project regularly coordinates *Open Content Projects*, projects that set a goal of producing a significant creative product (such as a short film or a game) *using* the software. These projects are funded through donations, grants, and pre-orders of the content being produced (such as DVDs), with the funds used to support developers and artists working together in a collocated space to produce the final product. (Note that these projects should not be confused with the movement of the same name that became the Creative Commons Project).

Open Content Projects provide a strong forcing function for addressing and improving software usability because the artists and developers are mutually dependent on one another to achieve the shared goal. These projects also serve as a significant morale booster for the entire community by giving them a clear, well-defined target, and a finished product that can be shown to, and appreciated by, the general public. A number of notable short films and games have already been produced by this model, with the software benefiting greatly in the process. While the 3DA project is the only project we are aware of that has such a process, it nonetheless provides a compelling model of how FOSS projects can be driven by real-world user needs in the absence of economic incentives.

Dedicated Usability / UX People

While the practices already mentioned involve developers and users, a number of projects we interviewed also had dedicated UX contributors. UX engineers and designers can sometimes face obstacles to joining a FOSS project [3,16]. However, in our study, we found that they are generally quite appreciated after they integrate with the project, as D4 indicates:

> *I'm quite happy and content to be the low-level developer, who says, "These things will be possible… [but] there are going to be hard, difficult interaction issues to tackle here." I'm quite happy for others to ponder those problems… I'm glad people are looking into improving [the user interface].*

After joining a project, the UX contributors often stated that they spend a significant amount of time simply educating project members about how to think about and practice usability/UX on a day-to-day basis. UX3 touches on this subject as she describes her goals in running design clinics at her project's primary conference:

> *I'd like those guys that I met at [our project conference this year], when they start their project and they think about adding 15 features to an application… [I'd like to think] that I have given them something that makes them stop for a moment and think, "Should I be adding this feature? Is it right for the product that I want it to sit in?"*

The UX members of FOSS projects, especially larger projects, echoed this basic sentiment: Their current goals are to make themselves known in the project's community, to make it clear how they can help the project, to educate project members about usability issues, and, finally, to provide usability feedback on the software. As an example, UX5 holds monthly meetings where she makes herself available to provide expert reviews on existing applications or proposed designs. Not only do these meetings provide immediate benefit to the developers, they also help remind developers that there is a need to consider usability in their day-to-day work.

From this survey of methods for discovering and addressing usability issues, we now turn to motivations for attending to these issues.

MOTIVATIONS FOR USABILITY

As we have mentioned, an oft-repeated maxim is that FOSS developers develop software to "scratch an itch" [21]. However, our study strongly suggests that while contributors may get *started* by scratching an itch, these "itches" quickly run out. Instead, our study indicates that social relationships, social rewards, and subtle social pressures are some of the most important forces to keep people involved in a project. These social factors also play a critical role in motivating individuals to address usability issues, as we describe next.

As an example of the strong social relationships that can form in these projects, NCC2 describes the friendship that has grown between himself and another project member as they wrote a manual for the DTP project:

> *You develop that sense of closeness. I mean, I feel like he's one of my best friends really, that I've ever had in my life. Which seems odd, because he's over there in Germany, and I'm over*

here in North America, and we've never met, and before a couple of years ago, I had no idea who he was.

These social relationships create an important glue for keeping a project together, as UX5 indicates when she describes why she continues working with the project:

'Cause I think it's important. DWE has a really great community once you actually get through the door. And I feel like I would be leaving, letting a lot of people down if I just disappeared, because there isn't anyone else to do this.

Not only do the social relationships keep people involved in a project, they also serve as one of the primary motivators for addressing usability issues on a day-to-day basis, as D8 describes:

I guess the reward for us is mostly getting more good input or feedback about BG. It is way more rewarding to have some guy like our reference artists, who are using your program on a regular basis, and do immediately give feedback… instead of having 10 people on the mailing list complaining that BG is not like Photoshop. In that sense, one person using BG is more valuable than 10 people using BG.

As D8 indicates, rich, high quality positive feedback from respected users is a potent reward for good design. D10 reinforces this point when describing why he would ask people in a university library for feedback on his software, which now has an installed user base of approximately 4 million people:

I just wanted evidence. I wanted to gather evidence that my program was usable. I had become so familiar with it, that I just wanted to know what it was like for other people to use it for the first time; what that first experience was like for your average person. Also, I just wanted praise. I wanted to show something that I accomplished to other people and have them acknowledge it.

Together, the points made by D8 and D10 summarize our findings on what motivates FOSS project members to attend to usability issues: After a certain point, the size of the user base is simply a number that serves as only a marginal reward; the true reward for developing well designed software in the FOSS community is praise and positive feedback from the user community. In an analysis of the discussion forum for an open source project, Iivari found users often thank and praise the developers for their efforts [11]; our data suggests that such praise can have important effects on usability. This finding stands in contrast to common assumptions that increasing the user base is a compelling motivator for addressing usability issues in FOSS development [1,16]. This result also has important implications for the design of HCI methods for this community, as we describe next.

IMPLICATIONS FOR FOSS USABILITY

Given the growing prominence of open software development, it is natural to ask how one can improve usability practices in this culture. Collectively, the results of our study, along with past research, indicate that the FOSS community possesses a culture of practice and value system quite distinct from that in which the field of HCI has developed. In particular, the close social relationships we found between developers and users are uncommon in closed software development, and do not naturally arise as they do in the FOSS community. Thus, in considering how usability practices can be improved in this culture, it is necessary to not only consider how they may be improved from within the community, but also how existing HCI methods and practices can change to better match this culture of practice. In this section, we consider both perspectives.

Improving Practices from within the FOSS Community

Our study indicates that the FOSS community, as a whole, already engages in a number of practices conducive to creating usable software. However, apart from Human Interface Guidelines (HIGs), there is little collective awareness of all the ways the various projects address usability. Similarly, project members often discount their own ad-hoc practices and don't recognize the value they provide in creating usable software.

Given this general lack of awareness, one of the most immediate ways the FOSS community can improve its practices is simply for it to be more aware of what others are *already* doing to improve software usability. One way to accomplish this goal is to create a catalogue of the techniques that help improve usability, along with instructions on how to get the most benefit from them. This approach has the advantage of conferring a degree of authenticity and legitimacy to the techniques, since they are derived from the community itself. As an example, the Open Content Projects have proven highly effective and could be emulated by other projects. Making this basic approach more widely known could spur other projects to adopt similar types of practices.

Reconceptualizing HCI for FOSS Development

Past research has noted that the openness of the FOSS community creates opportunities for the larger user base to participate in usability processes [2,16]. For example, one approach that has been suggested is to provide facilities that enable users to easily report usability issues as the software is used [16]. However, we note that any such approach must scale elegantly with the size of the user base, to avoid adding significant burdens to project members. Keeping this constraint in mind, the results of this study suggest that one promising avenue is to develop different tools for the different classes of users, where these tools align with the roles these users already play in FOSS usability. For example, one could imagine providing a rich set of feedback tools such as screen capture and annotation facilities, but only in versions used by core users (such as development branches). Doing so would enable these users to more effectively communicate usability issues during development, while avoiding the problem of information overload that could result if these capabilities were included in the official release. To address the problem of the representative-

ness of core users, data could still be collected from the larger user base, but in forms that are more amenable to automated summarization. For example, applications could be instrumented to collect usage data, as has been done with GIMP [26].

CONCLUSION

The results of this research challenge common conceptions of how the FOSS community perceives, acts on, and is motivated by usability. Most importantly, our research stresses the importance of the direct social relationships between developers and users in addressing usability issues on a day-to-day basis. These findings argue for the need to research new HCI methods that operate in the culture and value system of the FOSS community.

REFERENCES

1. Bach, P.M. Design information sharing across multiple knowledge systems in a FLOSS community. *Proceedings of iConference '09*, (2009).
2. Bach, P.M., DeLine, R., and Carroll, J.M. Designers wanted: participation and the user experience in open source software development. In *Proceedings of CHI '09*, ACM (2009), 985–994.
3. Bach, P.M. and Carroll, J.M. FLOSS UX Design: An Analysis of User Experience Design in Firefox and OpenOffice.org. In *Open Source Ecosystems: Diverse Communities Interacting*. 2009, 237-250.
4. Baldwin, C. and Clark, K. The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model? *Manage. Sci. 52*, 7 (2006), 1127, 1116.
5. Benson, C., Muller-Prove, M., and Mzourek, J. Professional usability in open source projects: GNOME, OpenOffice.org, NetBeans. In *CHI '04 Extended Abstracts*, ACM (2004), 1083–1084.
6. Fielding, R.T. Shared leadership in the Apache project. *Commun. ACM 42*, 4 (1999), 42–43.
7. Frishberg, N., Dirks, A.M., Benson, C., Nickell, S., and Smith, S. Getting to know you: open source development meets usability. In *CHI '02 Extended Abstracts*, ACM (2002), 932–933.
8. Gutwin, C., Penner, R., and Schneider, K. Group awareness in distributed software development. In *Proceedings of CSCW '04*, ACM (2004), 72–81.
9. Hars, A. and Ou, S. Working for Free? Motivations for Participating in Open-Source Projects. *Int. J. Electron. Commerce 6*, 3 (2002), 25–39.
10. Hedberg, H. and Iivari, N. Integrating HCI Specialists into Open Source Software Development Projects. *OSS*, (2009), 251-263.
11. Iivari, N. "Constructing the users" in open source software development: An interpretive case study of user participation. *Information Technology & People 22*, 2 (2009), 132-156.
12. Lakhani, K.R. and Wolf, R.G. Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. In J. Feller, B. Fitzgerald, S. Hissam and K.R. Lakhani, eds., *Perspectives on Free and Open Source Software*. MIT Press, 2005.
13. Mockus, A., Fielding, R.T., and Herbsleb, J. A case study of open source software development: the Apache server. In *Proceedings of ICSE '00*, ACM (2000), 263–272.
14. Müller-Prove, M. Community experience at OpenOffice.org. *interactions 14*, 6 (2007), 47–48.
15. Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., and Ye, Y. Evolution patterns of open-source software systems and communities. In *Proceedings of IWPSE '02*, ACM (2002), 76–85.
16. Nichols, D.M. and Twidale, M.B. The usability of open source software. *First Monday 8*, 1 (2003).
17. Nichols, D.M. and Twidale, M.B. Usability processes in open source projects. *Software Process: Improvement and Practice 11*, 2 (2006), 162, 149.
18. Oram, A. Why Do People Write Free Documentation? Results of a Survey - O'Reilly Media. http://onlamp.com/pub/a/onlamp/2007/06/14/why-do-people-write-free-documentation-results-of-a-survey.html?page=1.
19. Oreg, S. and Nov, O. Exploring motivations for contributing to open source initiatives: The roles of contribution context and personal values. *Comput. Hum. Behav. 24*, 5 (2008), 2055–2073.
20. Raymond, E.S. The Luxury of Ignorance: An Open-Source Horror Story. http://www.catb.org/~esr/writings/cups-horror.html.
21. Raymond, E.S. *Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.
22. Rooney, P. IBM Builds Dedicated Sales Channel For Red Hat, Novell Linux. 2005. http://www.crn.com/software/175002626.
23. Scacchi, W. Free/open source software development: recent research results and emerging opportunities. In *ESEC-FSE Companion '07*, ACM (2007), 459–468.
24. Scacchi, W., Feller, J., Fitzgerald, B., Hissam, S., and Lakhani, K. Understanding Free/Open Source Software Development Processes. *Software Process: Improvement and Practice 11*, 2 (2006), 95-105.
25. Shneiderman, B., Plaisant, C., Cohen, M., and Jacobs, S. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison Wesley, 2009.
26. Terry, M., Kay, M., Van Vugt, B., Slack, B., and Park, T. Ingimp: introducing instrumentation to an end-user open source application. In *Proceedings of CHI '08*, ACM (2008), 607–616.
27. Thomas, M.P. Why Free Software has poor usability, and how to improve it. http://mpt.net.nz/archive/2008/08/01/free-software-usability.
28. Twidale, M.B. and Nichols, D.M. Exploring Usability Discussions in Open Source Development. *Hawaii International Conference on System Sciences*, IEEE Computer Society (2005), 198c.
29. Ye, Y. and Kishida, K. Toward an understanding of the motivation Open Source Software developers. In *Proceedings of ICSE '03*, IEEE Computer Society (2003), 419–429.