Journal of the Association for Information Systems
JAIS

Special Issue

Characterizing the Dynamics of Open User Experience Design: The Cases of Firefox and OpenOffice.org

Paula M. Bach
Microsoft Corporation
pbach@microsoft.com

John M. Carroll
The Pennsylvania State University
jcarroll@ist.psu.edu

Abstract

We describe two cases of open user experience (UX) design using the Firefox web browser and OpenOffice.org office suite as case studies. Open UX design refers to activities and practices surrounding designing for the user experience in open source contexts. We analyze the socio-technical complexity of integrating UX activities into the two open source projects using activity awareness, a theoretical framework for understanding team performance in collective endeavors of significant scope, duration, and complexity. The facets of activity awareness are common ground, communities of practice, social capital, and human development. We found differences with the following dynamics in the two open source projects: community building, UX status in the community, decision making, and ways of sharing design information.

Keywords: *User experience, open UX design, Firefox, OpenOffice.org, design, FLOSS, open source, activity awareness, complex teamwork.*

* Michael Wade and Kevin Crowston were the accepting senior editors. This article was submitted on 15th October 2009 and went through one revision.

Volume 11, Special Issue, pp.902-925, *December* 2010

1. Introduction

Free/Libre/Open Source Software (FLOSS) project communities vary in their structure and the kinds of contributions made by participants. FLOSS communities attract developers who write code, but they also attract contributions of other kinds. For instance, participants localize software for different countries, submit various software issues to the bug tracker, and answer questions about using the software. Open source software communities, by sharing contributions from people of varying skills, have revolutionized how software is developed [49].

The socio-technical structures[1] of FLOSS have been investigated extensively [13, 17, 18, 26, 27, 32, 33, 36, 37, 39, 42, 43, 52, 53, 55, 56]. These studies have found that socio-technical structures are important because anyone interested in contributing to the production of a FLOSS project must learn to negotiate the structures in order to participate. Social structures involve the skills and procedures necessary for contribution. The technical structures include distributed development and coordination activities using email, inter-relay chat (IRC), discussion forums, and concurrent version systems (CVS).

Much FLOSS literature focuses on socio-technical structures that describe the developer community. As such, we know much about ways in which developers engage in the socio-technical structures, but little about other roles. The onion model depicts community structure in open source [3] (see fig. 1), and its concentric rings represent decision-making power with respect to contributions. At the innermost core, decision making is most concentrated. The core team consists of one or more developers who make decisions about the software's direction and contributions to the code base. The next three circles consist of developers. The committers have commit rights that allow admission of code into the main code base. Active developers regularly contribute to the source code repository, and peripheral developers submit code as patches, meaning their code does not immediately enter the main code base and must be first reviewed by the core team. The last two circles represent users. Active users contribute documentation, localization, support for other users in discussion forums, or bug reports. Passive users download and use the software without connecting to the community.

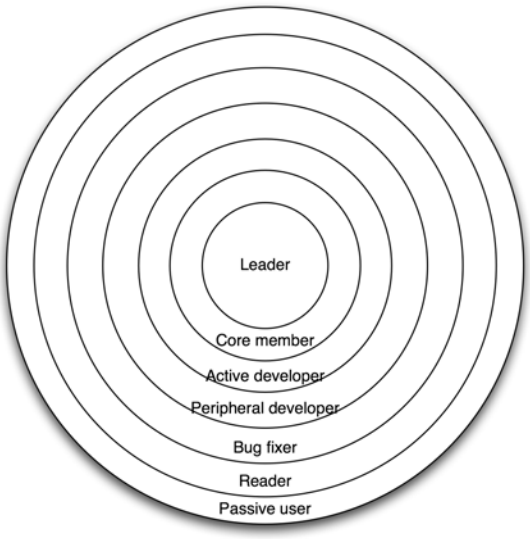

Figure 1. Roles in the Socio-Technical Structure of Open Source Projects

[1] We use the term 'socio-technical' to refer to social and technological aspects of an open source environment. As such the term does not refer to the socio-technical systems theory espoused by Trist et al Trist, E. and Murray, H. The Social Engagement of Social Science: A Tavistock Anthology, The Socio-technical Perspective. University of Pennsylvania Press, Philadelphia, PA, 1993. and others.

These socio-technical structures often pose problems for design because they are part of the open source developer culture and not the open UX design culture—a characterization which we explore in this paper. Open UX design practice refers to user experience design in FLOSS contexts. FLOSS projects are characteristically managed on the web and, therefore, much of the process is open to the public. In this sense of open, then, we are looking to articulate a new way of designing in the open.

Few researchers have undertaken a characterization of the socio-technical structure for open UX design. Hedberg and Iivari [30] developed a hypothetical model that includes design roles. Their model includes another level on the onion—a stacked onion—that adds a "human level" to the technical level. The decisions flow between the design (human level in the model) core and the technical core. See fig. 2 for the proposed model.

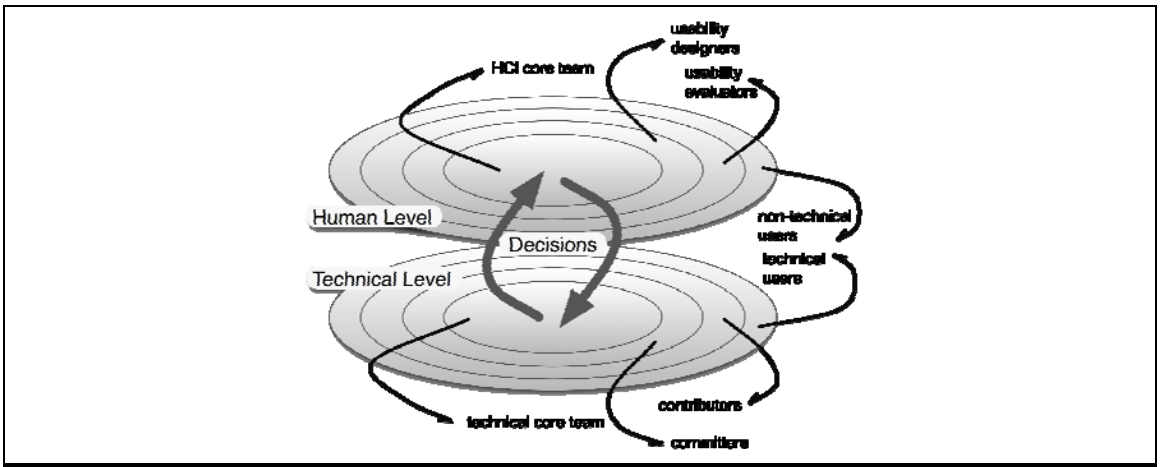

Figure 2. Proposed Socio-Technical Structure of Open Source and Open UX Design

Open source socio-technical structures have been studied and characterized in previous literature [1, 7, 18, 20, 22, 36, 37, 39, 42, 50, 55, 60], but open UX design has not. Given Hedberg and Iivari's proposed model, we seek to understand further the integration of UX in open source environments. We do so by characterizing two projects that have an established integration.

User experience refers broadly to the encounter people have with interactive systems. The goal is to design interactive systems so that they elicit a positive user experience. The HCI communities of researchers and practitioners have adopted the user experience goal, despite its definition being "vague, elusive, and ephemeral" [28]. User experience designers bring several different methodologies and theories to their practice. While user experience design practice includes other design approaches such as user research [40], interaction design [59], and usability engineering [51], among others, it also strives to incorporate experiential and affective dimensions in understanding the user. These design approaches combine in various ways, depending on the audience and the product, to ensure a positive user experience. Previous research has found that sharing UX information is challenging and complex [6] and that UX practices tend to be marginalized in organizations [10]. While the integration of software development and UX practices remains unfulfilled [58] in closed source development, we cannot assume that similar approaches and attempts to find common ground with the two disciplines of UX and software engineering bear any similarity to open source environments, because the socio-technical structures of open source and closed source development differ [24].

Most open source projects do not explicitly incorporate UX design (UXD); however, some projects do have particular UX strategies in place. Some quite prominent open source projects have begun to take UXD more seriously and to articulate and enact explicit UX strategies. For example, the Firefox web browser and OpenOffice.org office suite, the two cases described here, employ UX practitioners that are responsible for UX strategies.

Open UX design is a recent phenomenon. FLOSS development is clearly different from traditional software development approaches. Thus, it is a question of whether and how existing UX approaches apply to open UX design. FLOSS developers find bugs, submit features, write code, review code, and coordinate code integration in fast iterations that are released often [17]. Developer work is merit-based, and developers who are highly skilled and knowledgeable hold leadership positions and make decisions while they gain trust from other developers [54]. Any developer or contributor can find a FLOSS project to work on as long as he or she adheres to the project's social and technical structures [20]. Such characterizations of open UX design, however, are rare.

The purpose of this paper is to investigate how activities from one domain, user experience design, and another domain, FLOSS development, integrate given the potential mismatch between their socio-technical environments. When combined, the dynamics of each domain have potential for disaster, wrought with conflict, misunderstanding, and distrust. Yet, despite conflict, misunderstanding, and distrust, two open source projects have begun the process of integration. Although they have by no means worked through their problems, this point in time illustrates important dynamics. Therefore, our contribution entails characterizations of UX in open source environments. As such, we found that building community, the status open UX design has in the project community at large, the way UX design decisions are made, and how UX design information flows throughout the open source project environment are important socio-technical structures for strategizing UX design in FLOSS. The motivation for this work is to fill the gap in the broader open source literature about socio-technical characterizations of open UX design.

2. Challenges to and Opportunities for Open UX Design

The literature exploring issues related to understanding how UX design (UXD) fits into FLOSS development environments articulates challenges and opportunities for characterizing open UX design. The background literature exploring the challenges and opportunities in open UX design is limited, however. What follows are the seminal studies first investigating usability issues in FLOSS.

Nichols and Twidale [46] suggested that user experience activities in open source projects may be restricted by the software development process, lack of average user involvement, and the culture of open source communities. They articulated the challenges and presented them as nine barriers to open source usability shown in Table 1 below, where we've grouped them into three types. In later studies, Nichols and Twidale explored some of the approaches to FLOSS user experience suggested above. These solutions explored the possibility of bringing users together to participate in the user experience of the project [47] and to collect usability data from the software [46], and how to coordinate and evaluate user interfaces in FLOSS [62].

Table 1. Barriers to UX Design in FLOSS

Types of Barriers	Barriers
Developer perception	Developers perceive users as equally technical or annoyingly stupid
	Developers perceive usability problems as trivial and unchallenging
	Developers perceive usability problems as functionality problems
	Developers value power over simplicity: complex software is more powerful but can be difficult to use
Community integration	Difficult for usability professionals to integrate into the open source culture
	No resources for high quality usability work
Process constraints	Difficult to innovate on process because mental models already established from closed source software
	Function-centric software that anybody can add to can result in 'feature bloat'

Other research also described and identified user experience issues in open source software. Under the umbrella of quality, Hedberg et al. [29] reviewed literature on open source usability and quality. Their review captured the main issues: developers who use the software are different from non-technical users; the FLOSS development processes are anything but user-centered; usability specialist participation in FLOSS is rare; current FLOSS usability practices happen too late in the process; traditional usability methods conflict with the spirit of open source; and the distributed and collaborative nature of FLOSS development challenges integration of usability. Another analysis highlighted the problems with integrating traditional methods like user-centered design into FLOSS development [63]. The authors explained that the distributed, collaborative nature is deeply tied to the technological infrastructure, and coordinating user experience activities adds to the complexity of the ecology. These analytical studies exposed challenges for UXD in FLOSS, particularly with respect to integrating practices not familiar to software development, in general, and FLOSS development, in particular. These process challenges are further made complex by the strong developer culture embedded in FLOSS environments and the difficulty with which user experience design activities would integrate into such environments. This exposes a worldview challenge, where user experience designers approach problems vastly differently from FLOSS developers.

Validating the analytical studies, empirical studies confirmed the different worldview challenge suggested by the analytical studies [2], [62]. Andreasen et al. [2] concluded that developers have a limited understanding of user experience and that more research needs to occur to understand the sociological patterns in FLOSS communities so that user experience can be better integrated. Twidale and Nichols (2005) suggested that usability experts need a toolkit that supports handling “usability bugs” such as including metadata and structured information, and including a designated design area that supports discussions and prototyping. A study of user experience practices in Mozilla and GNOME open source projects uncovered the following issues: complexity management of usability bugs, user involvement in bug reporting, and using interface [48]. Based on these findings, Twidale and Nichols suggested technological opportunities for integrating user experience practices in open source communities (2006). They reported that design discussions in Mozilla occur in a weblog. This solution has several advantages over a bug tracking system meant for functionality bugs: 1) designers can tweak designs, although this may happen too quickly for important UX people to participate; 2) the blog is more open to many designers than bug trackers; 3) because blogs are (at this point) smaller, things are easy to find; and 4) blogs support conversational workflow better than bug trackers. Because of these factors, they suggested again that a better usability bug tracking system is germane for the integration of UX practices. Although Twidale and Nichols explored details further, they have not suggested a complete set of design features for such a usability bug tracking system. This study exemplifies how user experience practices were assimilated with developer practices, resulting in sub-optimal conditions for user experience activities because incorporating best practices with respect to user experience design was too challenging.

Another empirical study investigated an open source company, which the researchers defined as a “software development company that tries to gain competitive advantage of OSS” [38]. In the case study, which included interviews of two user experience professionals, researchers found that the UX professionals gained insight from the real user feedback available in open source contexts. Another observation made in the study is that with the FLOSS community helping with code development, more resources can be attributed to usability and user interface design. This study is qualitatively different from the above studies in that it outlined some benefits of integrating user experience activities into FLOSS development rather than exposed challenges.

Bach et al. [5] explored the social and technological complexities of integrating UXD practices and increasing UXD participation in an open source project hosting website community. The study argued that in order to open up the open source environment and make it more welcoming, the design solution should foster ways to build trust between developers and designers, provide opportunities for designers to show merit and for developers to understand designer merit, and provide tools that support UX design best practices The study proposed a socio-technical design solution with features that supported UXD activities such as user research, a to-do list, design iterations, and user evaluations.

With challenges such as differences in worldview, ineffective technologies for design work, and choices for UXD methodology in FLOSS come successes such as community building and user feedback, and opportunities for UX professional participation and support for UXD activities in FLOSS projects. The challenges and opportunities articulated in the literature point out where current FLOSS socio-technical structures support developer activities more than UXD activities. Thus, characterizing open UXD strategies in open source environments is important so that we may understand further the open UX design phenomenon and begin to articulate strategies to build, support, and sustain socio-technical structures in open UX design.

3. Characterizing Open UX Design

To characterize open UX design, we employ a theoretical framework that affords an analysis of how open source projects have integrated UXD activities. Because open source environments pose challenges for UXD practice due to differences in culture, technology use, and community involvement, we propose a theory that operates at the activity level. The activity level is appropriate for analysis of UXD integration strategies because *practice* occurs at the activity level. Therefore, we capture the dynamics of everyday practice. This uncovers activities of open source contributors of all types, already are part of an open source community, and UXD contributors, who may or may not already be part of the community. When UXD activities are integrated, both during and after initial strategizing has taken place, the community must adjust and *be aware* of UXD activities in order to make the best use of their value.

To date, the fields of Computer-Supported Cooperative Work (CSCW) and Human Computer Interaction (HCI) have worked to theorize awareness [57] in research investigating collaboration and teamwork. In general, such theorizing investigates awareness of synchronous work such as who is present, where one is pointing in a shared workspace, where a co-worker is looking, or what a co-worker has recently done [12]. In the open source context, however, much work is accomplished asynchronously using tools such as email lists or discussion forums for communication and concurrent version systems for coordination [68]. Carroll et al. have proposed a theoretical framework that "constructs awareness at the activity level to theorize awareness as a collective achievement with a developmental trajectory," and they "refer to the mutual awareness of partners in a shared activity of significant scope and duration as activity awareness" [12]. Given that activity awareness theorizes at the activity level with a focus on teamwork, it serves as a useful lens to help characterize the teamwork activities involved in integrating open UX design into an open source project community. In the next section, we further unpack the activity awareness framework and describe how we apply it in the context of open UX design.

3.1 Activity awareness

Given the above discussion, an appropriate theoretical framework for characterizing open UX design is one that provides constructs for making sense of everyday activities in and across the different types of open source contributions so that different aspects of UX activities can be analyzed. The aspects include communication, practice, learning, information sharing, meaning, obligations, and interactions.

The theoretical framework activity awareness characterizes four facets (common ground, communities of practice, social capital, and human development) for understanding and describing group awareness of activities – that is, how groups maintain awareness of, effectively regulate, and continually improve their own activity [11, 12]. The activity awareness framework was motivated by an approach to complex teamwork, work where teams cope with "ill-structured problems in real-world contexts of high uncertainty," where the focus is on shared activities [11]. This approach allowed for a conception of human behavior in team settings that addressed the question of what groups need to share in order to work effectively, both individually and collectively. Activity is defined as "substantial and coherent collective endeavors directed at meaningful objectives" [11]. Activity awareness is premised on the idea of awareness: that, in order for teams to work effectively, they need to know what relevant information their collaborators know, and what they expect, as well as their attitudes and goals. They need to know what criteria collaborators will use to evaluate joint outcomes, the

moment-to-moment focus of their attention and action during the collaborative work, and how the view of the shared plan and the work actually accomplished evolves over time [11] .

Thus, the concept of activity awareness prioritizes well-informedness in collaborations with consequential goals. This prioritization is founded on activity theory [8, 9, 45] but carries forth a further articulation of the concepts in activity theory to emphasize that group work is "activity-oriented, praxis-based, relative, and dialectic" [11]. Activity theory [21] describes mediating effects of collective activity, where members' shared understanding of roles, community values, division of labor, tools, task resources, and other artifacts mediate social interaction. Activity theory highlights motivation and shared goals by recognizing and addressing breakdowns, social conflicts, tool weaknesses, and resource shortcomings. In the activity awareness framework, the mediating effects of activity theory are analyzed via the sub-processes of common ground and communities of practice. As such, further articulations of activity theory draw on four facets: common ground, communities of practice, social capital, and human development.

Because communication among team members and among different teams is a foundational skill, the facet common ground [15] articulates a protocol for dynamic monitoring, testing, and signaling of whether knowledge and beliefs are shared and, therefore, setting a basis for understanding. Teamwork requires the coordinated efforts of people to accomplish goals. Communities of practice [66] enact coordinated efforts together with shared goals, values, and practices specific to a community. Voluntary membership in a community needs powerful social mechanisms to support sustained participation through challenges, both through problem solving and conflict. Social capital [16] is the acquiring of social benefits of past interactions in order to reduce conflict and other risks in future interactions. When teams interact during collaboration, they plan, negotiate, and coordinate over stretches of time. They solve ill-defined problems of high uncertainty and with consequences. In such a collaborative environment, people change. Thus, human development [64] is when people solve open-ended, complex problems and evolve not only their skills, but also the skills of members and teams, resulting in innovative behavior and decisions. Because the activity awareness framework includes facets that articulate the dynamics of team interaction, it is a good framework to analyze the dynamics of open source project team activities and, through analysis, offers a characterization of open UXD. We further articulate the details of each of the four facets of activity awareness in a later section alongside the analysis of each case through the lens of each facet.

4. Approach to Research

To characterize open UX design, we employed qualitative approaches to uncover deeper understandings of everyday practice. Our methodological approach was similar to a virtual ethnography, [34] where we observed online communications either as they unfolded or as a discussion document and analyzed design artifacts such as specifications and design spaces including UX wikis and blogs. Here, we present descriptions of our observations as a cumulative account of online activities. A key aspect of studying design, in general, is gaining access to design teams and their materials. We selected Firefox web browser and OpenOffice.org office suite as cases because they have for at least two years had UX strategies in place and their design materials are openly available on the Internet. In the case of open UX design, access to some of the team's communication was easily gained via email distribution lists, while other communication was not available because it may have occurred via inter-relay chat, telephone, or face-to-face. As such, we've studied the team's design discussions only through asynchronous communication channels.

We studied the cases informally during fall 2006, followed by a more intense study period over 12 weeks during the summer of 2007. The study consisted of email distribution list postings and observation and document analysis of each project's corresponding website and blog. We conducted one open-ended interview at an open source conference in October 2007. During the summer 2007 study, we recorded field notes daily capturing interesting quotes from online discussions and wiki and blog postings. The study continued informally through to fall 2008. The informal observations (fall 2006 and fall 2008) were monthly recordings of activities on email lists and websites. For each of the cases, we describe the community and its approach to integrating UX into the community, then

describe events of interest for understanding how open UX design occurs. Last, we offer an analysis of the events using the activity awareness framework. We chose these events because of their richness and how they exemplified different strategies for UXD integration.

Both cases are of a hybrid nature. Hybrid software development organizations consider business models, economics, social structure, community, and ideology. Fitzgerald [25] describes such a hybridization as a transformation that can be achieved through a "balance between a commercial profit value-for-money proposition while still adhering to acceptable open source community values." Our intent is not to define whether the two cases presented here are hybrid, but whether they have attributes of both commercial and open source software development organizations and, as such, can shed some light on strategies for UX integration in software development environments that sustain attributes of both open source and commercial software development.

5. Two Case studies of Open UX Design: Firefox and OpenOffice.org

We describe the two cases of open UX design. These descriptions include vignettes that illustrate the dynamics of integrating open UX design in the two FLOSS environments. We then analyze the various vignettes through the lens of activity awareness and examine each case for articulations of the four facets, common ground, communities of practice, social capital, and human development.

5.1 Firefox

Firefox open UX design started in 2006 when Mozilla hired a UX director to lead design. In the next year, the company hired two more UX practitioners. While much of the design process is open and available on the web, some decisions were made behind the scenes by the management teams, board of directors, and core development team. The description of the development process reported here is based on an interview with the UX director[2] conducted in October 2007 and analysis of the online documents and conversations. A wiki document outlining the planning and design for Firefox 3 was created in late May 2006. This wiki contained a requirements document and feature list, among other information.

The community was encouraged to participate in the planning and design through three discussion groups. The three groups differ in their content and activity. The most active of the three groups is the Firefox development group. The group discusses development issues such as troubling bugs, ongoing problems, and new proposals for bug fixes or new features. Another discussion group we investigated was the planning group list, where the community discusses planning for future releases. The last group we observed was the usability list. During the period May through August 2007 we read daily messages and recorded field notes of discussions surrounding user interface components. As such, these discussions were more likely than other discussion groups (e.g. developer group) to influence the user experience. During summer 2007, we observed 1,529 messages in the development discussion group, 961 in the planning discussion group, and just five in the usability discussion group.

The development team and the UX team participated in the discussions addressing concerns among team members and with the community of users. The feature list was continually updated based on the discussions in the list, and at various points "bugs" were created to initiate work on the features. Depending on the complexity of the work, a feature requirements document was created. FLOSS community members tracked all work on a specific project in a bug tracker. Traditional bugs are work items, or errors in the system that need to be corrected. But also bugs can also be new features or any other tasks that affect the code base. Much discussion, including design decisions, also occurred in Bugzilla—the bug tracker used for Firefox development. As tasks were completed, the bug was closed and status marked as complete in the requirements document.

Often, new features were discussed in the development discussion forum. For example, a lively discussion from May to September 2007 took place over a proposal to change the location bar in the following two ways:

[2] The UX director is now the director of front-end development, user experience, and product delivery at Mozilla.

1. *Remove the favicon from the URL bar. We want to make the URL bar totally trusted, and that means not allowing sites to control parts of it to spoof locks or things like that. We can either remove it entirely or replace it with a generic page icon/folder icon/whatever under our control.*

2. *Change the U RL bar so that everything except "Public Suffix + 2" is greyed out. If the URL bar is focussed or hovered over, the colour switches back to black throughout. This should be possible using CSS only. The "greyed-out" colour is a pref; people who don't like this feature can set it to "black".*

Following a review of the prototype with the UX and development leads, a Firefox developer put forth the proposal to see how the community would react to such a change. The motivation for the proposal was security-based and suggested providing the user with information about "who they're dealing with online," according to the Mozilla security developer (who is different from the developer who initially proposed the change). The UX lead summarized the discussion about URL highlighting and entered it into the wiki. This change, however, did not make it into the requirements document and, hence, Firefox 3, because it was unclear how much highlighting would help the user. However, developers and users posted different mockups for review, and one of the Mozilla UX practitioners suggested that even if they had an eye tracker available, reading highlighted text would probably be only milliseconds quicker when parsing the URL to determine if it was familiar.

A discussion in Bugzilla about information in the security tab within the preferences dialog occurred about how to present security information to users because information in the Firefox 2 security tab dialog was too technical. The lead security developer and two other developers submitted patches to a redesign, the UX director conducted a design review, and feedback from five other developers/users guided the design until another bug was created titled "Clean up Security Page Info visuals" to address the layout. Some discussion ensued, patches were proposed, and after the final User Interface review, both of these bugs were closed and, thus, considered fixed.

At first glance, UX design is not easy to recognize. Design work is carried out in discussion lists, bug trackers, and requirements documents, and there is no obvious single design space. Code, on the other hand, exists in repositories and is easy to download and work with. A developer can download modules of the code base and work on patches, but UX designers can't download various designs from a central repository and work on iterations. A unique aspect of open UX design is the participation of the community in the design and development. The Firefox community consists of about 40 core developers,[3] 100 daily contributors, 1,000 contributors, 10,000 nightly testers, 100,000 beta testers, and 30 million daily users. And although not all members contribute, or contribute evenly, the UX team has a considerable amount of information to integrate into UX design.

When interacting with community members and weighing their suggestions and feedback, the lead UX director considers two different philosophies for how to interject UX knowledge into the community. The first philosophy is to be the expert. In this approach, the Firefox UX practitioners are experts, and they know what is better for the user experience, just like developers are experts about code. The other approach is to provide research and data, to back the UX design with science. Commenting on these two approaches, the UX director states:

> *What needs to happen is that we need to say that our opinions are rooted in observational science, perceptual science, that there are foundations for our expertise. And that we need to build credibility with these kinds of expertise, but we should be given a free rein to play around with things. And we should be trusted a little more.*

On one hand, the UX director believes designers should be trusted to come up with appropriate designs with which the community can experiment. On the other hand, he believes designers should provide a certain amount of rationale based on science. Of course, these two approaches are not

[3] These numbers are from spring 2007.

mutually exclusive. They both occur to some extent in the community depending on the designer's reputation.

To introduce UX information such as design rationale or perceptual science, one of the Mozilla UX practitioners maintains a blog about UX to share information with the community. For example, one blog entry about quantitative design talks about cognitive performance modeling and why "your mom"[4] is not statistically significant, or more formally, why it could be a mistake to rely on cursory single cases, or worse, imagined single cases. According to the UX director, the downside of providing data all of the time for design decisions is that the community will be afraid to commit to changes unless they are backed up by science. What the UX team should be striving for, the director says, is to have the community accept that some design changes can be playful and open for discussion. But he also states:

> *[The Mozilla community is] highly motivated and users care more. Paranoia and nervousness to protect the user experience result in conservatism.*

Given the complexities surrounding open UX design, awareness of UX activities in and by the Mozilla community is essential for understanding the characteristics of open UX design.

5.2 OpenOffice.org

We collected the OpenOffice.org case study data primarily from May 2007 to November 2008. Data collection included observation of the following OpenOffice.org UX online activities: five email lists, website, blog, and wiki. In addition, we followed up on some discussions in the bug tracker. Finally, data also came from articles published by a member of the UX team. OpenOffice.org (OOo) is an open source office suite derived from the StarOffice suite, which was developed by StarDivision and acquired by Sun Microsystems in 1999.[5] Sun released the source code in July 2000. In January 2007, the UX project was launched. OOo consists of several projects surrounding the community development of the office suite product. Projects begin as incubator projects and can move to "accepted" status with evidence that the community supports the project. Categories of projects include product development, helping users, promotion, and language support, among others.

The UX project is one of several product development projects. Project leads have a vote in the overall OOo decision-making process. A community council and an engineering steering committee govern the OOo community. Originally, before it was an official project, the UX project began via a new mailing list intended to gather a community of user experience experts wanting to help improve OpenOffice.org. To that end, the UX team established a user experience community infrastructure that includes a user experience home page on a sub-domain (ux.openoffice.org), a wiki, five mailing lists (cvs, commits, discuss, issues, and request), inter-relay chat channel, and user experience blog. Since the community infrastructure was deployed, the project has seen a sharp increase in UX expert participation [44]. The UX team consists of six Sun employees and other community members who have an interest in UX.

The OOo UX website offers detailed instructions for how to become a member of the UX team. To become a member, a UX-interested person must register, request a membership on the team, make introductions, explore the UX resources (includes usability studies, literature, specifications, and so on), and finally, pick one of the many issues on the to-do list. During the time of the study, 38 UX team members were listed on the UX wiki. The members ranged from Sun UX practitioners to OOo users, interaction designers, a medical doctor, developers, and students. The UX website provides a quick link to a to-do list which is compiled by the UX lead and other members of the UX team. The list includes links and descriptions of issues categorized, for example, by release version, number of votes, and expert talks with customers. Issues are linked to the bug tracker and, if applicable, to a specification. The OOo UX project had one active discussion group during the summer of 2007 (when

[4] In open source communities, developers often justify UX design decisions based on how their mom or their grandma might easily use the software.
[5] In April 2009, Sun was acquired by Oracle.

we made daily observations) and has subsequently added more. During the May through August 2007 time period, 431 messages were posted on the UX discussion list. The messages consisted of user interface proposals and discussion of usability bugs and how to fix them.

Community building for the UX project was deliberate. The UX lead wanted to change the project's status from incubation to accepted. An incubation project on OOo is one that has not been fully accepted by the community. A project that is a testing ground for ideas is categorized as an incubator project and is governed by less strict rules. Such projects may later make it into the accepted category. As such, the decision to move the UX project from the incubator category to an accepted project was ignited by a post on the discuss list with the subject title "UX – the secret project…." Before the project came out of incubation, it was only discoverable via search because it was not listed on the projects page and, therefore, was difficult to find. The UX lead worked with the community to assess the project's usefulness and to establish it as an accepted project. He posted a message asking developers what they expected for resources and how they wanted to collaborate with the OOo UX community. Nobody responded. However, a few weeks later, another member of the UX team posted a survey to the OOo UX community. The goal of the survey was understanding the community better in order to change the project's status and learn about the UX community in the following areas: IT infrastructure usage, satisfaction level, and critical gaps to close.

The results indicated that the UX community (in July 2007) mainly consisted of users and a few UX practitioners; needed tools and space for collaborative design; used the mailing lists for two-way communication; and wanted more closure from the discussions on the mailing lists, that is, more decision making. One survey participant noted that the UX portal was missing crucial information such as process and usability information. And another participant, dissatisfied with being part of the community, noted, "Seeing user-experience issues actually implemented in the released software – it just takes way too long and takes too low a priority." In addition, a comment about information flow noted blockages: "Huge barrier to entry. Discussion on mail list is just opinion; next step is to write a complex spec. Developers then make the final decision." While the Sun UX team took steps to build the community, barriers existed both within the OOo UX community and the larger OOo community.

The top UX Calc (spreadsheet application) issue for April 2007 was a bug reported in fall 2002. The top 20 voted-on-issues were listed on the UX Calc to-do list and ranked by number of votes from users. The UX issue list was taken from a second quarter review of Calc posted on the main OOo wiki. In the comments section of the bug tracker, users discussed the behavior of the bug and specified how the application should work given the task. Five years later, a patch was proposed, but it lacked full functionality for the task. Two users posted descriptions of the patch and one posted his specification on the UX list for feedback. In this case, the users specified how they thought the interaction should work. The discussion continued for two more threads on the UX discuss email list with the user who posted the specification and a contributing developer.[6] It is unclear if the patch will be reviewed again and developed according to the specification created by the user, and submitted to the issue tracker for the core team to commit to the code base. Alternatively, the issue could be pushed by the UX lead to a core developer. Core developers are often available immediately after a release, but developer resources begin to be used up throughout the release cycle. Although the Sun UX lead and co-lead have not been involved in this bug fix, UX leads must be involved in creating specifications for new features, if not for reviewing bug fixes, as was evidenced by the five-year old bug.

The OOo UX team posted blog entries about important design decisions. For example, the team was working on a new design for adding editing notes to Writer, the word processing application in the OOo suite. A team that included two UX team members, two developers, quality assurance testers, and a document specialist worked on the feature. A first blog post included a step-by-step rationale for design decisions. A later blog post responded to complaints about the color palette used for the notes. The post explained why color is important for accessibility (e.g., color blindness) and information visualization. These blog posts provided opportunities for community learning.

[6] Contributors are developers that are users, but not part of the core development team employed by Sun or otherwise nominated and voted into the core development team.

Given that the OOo UX team is focused on community awareness, activity awareness is important for understanding how UX activities are integrated into the larger OpenOffice.org community.

6. Activity Awareness Analysis

In this section, we present a general overview of each facet (common ground, communities of practice, social capital, and human development) followed by an analysis of vignettes through the lens of a given facet for each of the two cases.

6.1 Overview of Common ground

Common ground is a communication protocol for checking and indicating shared knowledge and beliefs. Clark (1996) states that two people converse through joint action. During conversation, participants reach common ground through their ability to coordinate the source of their joint action. Common ground is, therefore, the set of knowledge, beliefs, and suppositions that the people conversing believe each other shares. Conversation can only progress successfully if people establish and maintain common ground. This concept is particularly critical for multidisciplinary teams with differing knowledge sets and disciplinary perspectives. In addition, distributed groups have to continually work at and monitor common ground; they cannot ever take it for granted the way that face-to-face teams sometimes can. In teamwork, communication is important because successful collaboration depends on joint activity. When two people who are communicating reach common ground, they show, by linguistic signaling, that they understand each other, and thus, have shared knowledge and beliefs. This shared understanding makes interactions more efficient. Clark and Brennan contend that understanding can never be perfect, but linguistic signals can indicate how conversation continues to build toward a mutual belief such that "partners mutually believe that the partners have understood what the contributor meant to a criterion sufficient for current purposes" [14]. For example, Alan and Barbara are building and verifying the mutual belief that each knows that Barbara doesn't have a car.

Alan: Now, -um, do you and your husband have a j-car
Barbara: - have a car?
Alan: Yeah
Barbara: No -

Mutual belief occurs when Alan knows that Barbara knows that he knows that she doesn't have a car. After the first utterance, Barbara demonstrates that she acknowledges Alan, and later answers the question demonstrating that she has understood him. A more efficient interaction occurs when Alan displays that he understands Jon by finishing his sentence.

Jon: you know, he's just gonna - -
Alan: yeah

In the above exchange, Jon understands that Alan will understand without him completing his utterance. As such, potential collaborators look for exchanges like the one between Jon and Alan because they are more efficient. In a successful integration of open UX design, then, we would expect to see more rather than less efficient interactions, thereby indicating shared knowledge and beliefs at the intersection of development and design.

6.1.1 Common ground in Firefox UX

In the location bar discussion, both the security lead developers and the UX director provided summaries of the discussion to check that information was being understood appropriately. Also, the UX blog post about quantitative design provides a mechanism in the comments section where the UX team can see how community members are sharing common knowledge and beliefs, if any, or where breakdowns might occur. Perhaps the biggest breakdown in common ground, as described by the UX director, occurs when developers don't appear to understand the knowledge base of the UX practitioners, and this gap requires the UX people to work extra hard to be understood.

6.1.2 Common ground in OpenOffice.org UX

Although UX has an established presence as a project in the OOo community, the multidisciplinary nature of all projects associated with OOo presents challenges for common ground. For example, with the initial Notes blog, a UX team member posted twice, first explaining rationale for Notes, and the second explaining about colors. The need for the color explanation indicated that some members of the OOo community did not know about accessibility issues with color. This demonstrated a shared lack of knowledge about accessibility among members other than UX practitioners. Furthermore, the comment about lack of decision making and no closure on discussion is the result of the high cost for achieving common ground with electronic, asynchronous (e.g., email lists) communication [15, 19]. Email list participants simply have to work too hard to reach common ground on issues. This situation is complicated further by the global distribution of members. Although English is the language used for discussion on the UX lists, it is not always the native language of its discussants.

6.2 Overview of Communities of Practice

Communities enact activities that they share through practice. These activities are specific to the community members who share a tacit understanding of how to participate in the community. Developers wishing to join an open source community must understand how members enact activities and figure out social practices [20]. This poses problems for UX designers in open source communities, because sharing practices with developers involves a process of enculturation: learning a rich set of moves and expectations, a variety of signals that members may not even be able to readily articulate but which they regularly and fluently enact. When UX practitioners join a FLOSS community of practice, they must achieve a high level of awareness – they must know and recognize they can do the same things the other members do. Yet this awareness is further complicated because UX practices and development practices are not the same, an issue we discuss further below.

The activity awareness framework conceptualizes communities of practice as a “rich and more narrowly-scoped foundation” [11] for collaboration, more directly focused on joint activity than common ground. While common ground provides a foundation for shared knowledge and beliefs, communities of practice provide a foundation for sharing “praxis-domain-specific ways of thinking, organizing roles, and doing work” [11]. Communities of practice, then, enact social configurations consistent with a work domain where identity within that domain is built and maintained through problem solving. A personal investment in goals, practice, and values results in the collective investment of sharing activities. Thus, performing complex, collaborative, open-ended activities requires communities of practice.

When team members share a domain, they share activities and knowledge particular to the domain. Contributors to open source projects vary in their background knowledge and skills. For example, a skilled developer might have software architecture knowledge, whereas another developer might have limited skills in the programming language used to develop the software application, but nevertheless can submit a patch. Also, a skilled UX designer might have research and prototyping skills, whereas a user can describe in detail a confusion that occurred while using the software. These contributors share the domain of software development. In this domain, teams collaborate to produce software. Slippage occurs, however, with defining the boundaries of the domain, and therefore, the boundaries of communities of practice. While activities within a community of practice vary depending on role, different roles bring different knowledge to the domain, and as such, conflict may arise. We have seen such conflict in the literature when open UX design and FLOSS development domains come together under the larger domain of FLOSS community software design development. This is the conflict we seek to understand and use to help characterize open UX design. As such, when conflicts arise, communities of practice can work together to solve problems, which, in turn, sustains and enhances the community of practice.

6.2.1 Communities of practice in Firefox UX

The Firefox UX practitioners have negotiated the socio-technical structures by integrating their activities, for example, design reviews and rationale for changes into existing structures. Firefox UX

practitioners work in the bug tracker to monitor and guide the design changes and provide design reviews for final changes before a bug is closed. Furthermore, the UX team provides research-based rationale along with opinions. In the location bar discussion, a Firefox UX practitioner posted a link to a study exploring how users responded to toolbars with information about phishing and the legitimacy of a website. Traditional UX practices have been adapted to the project's socio-technical structure, and to some extent, developer practices have been adapted to accommodate UX design, for example, where developers ask for a design review. As such, the entire community, by adopting new practices, moves toward an open UX design community of practice.

6.2.2 Communities of practice in OpenOffice.org UX

Within the OOo UX community, implicit understanding of practices has not been achieved. Evidence of this is the need for guided direction in how to participate in and navigate the community. Furthermore, one survey participant had difficulty in ascertaining the general processes for achieving a good UX and finding usability methods used by the UX team. A gap exists even within the UX community. While the Sun UX team members are trained UX practitioners, not all of the members signed up for the UX team are knowledgeable UX practitioners. While users and developers can provide helpful feedback, their lack of understanding of UX activities, in general, results in the gap in practices. The community is further divided through a lack of understanding of FLOSS UX activities. While UX practitioners outside of FLOSS may share practices with other communities of UX practitioners, coming to FLOSS UX is a different kind of practice that cannot be understood through practices in non-FLOSS environments, for example.

The OOo UX community of practice participates in some activities enacted by the larger OOo community. As such, the communities of practice have some overlap. For example, the UX lead must participate in new feature specifications, and UX team members are active in bug tracker discussions. Different email lists roughly map to the different kinds of activities that occur on the OOo UX project. While these lists differentiate several communities of practice being enacted in many different sub-projects, the lists are open for anyone to join. Participation, however, is not a consequence of openness. A barrier to discussion carries over from the inability to cross communities of practice because even though participants may be lurking, they may lack the understanding needed to be full members of the community of practice associated with the activities on any given list and, therefore, be on the periphery. On the other hand, lurking is a good way to learn how to participate and gain legitimacy in the community [41].

6.3 Overview of Social capital

Complex teamwork requires successful interactions. When continued beneficial interactions build trust among team members and other networks in a social good, teams overcome adversity. These favorable interactions moving toward a persistent social good build social capital [16]. Open source developers build networks of social capital to help them solve problems. UX designers have a tougher time engaging in open source projects because building trust, social networks, and beneficial interactions with developers can be challenging. Challenges arise because achieving a level of trust and building social networks and beneficial interactions requires that one is already a member of the community of practice.

In order for developers and UX designers to build social connections, both within and outside of the teams, they must act cooperatively. When they act cooperatively, they establish a generalized reciprocity, but this is difficult to attain when the other members of the community do not yet respect an outside member's ability to perform and participate in ordinary activities of the community. Reciprocity norms, trust over time, mutual understanding, and shared values make cooperative action possible. As such, social capital refers to the mutual exchange of norms and values that serve collaboration. This exchange of values and norms "softens the consequences of inevitable differences in opinion, conflicts and even outright misbehavior" [11].

6.3.1 Social capital in Firefox UX

The Firefox UX team experienced frustrations regarding conservatism with design explorations. As an

example, in the location bar thread, one of the Firefox UX team members empathized, “I understand that you (and likely a minority of other users similar to you) will *hate* these changes.” The user responded, “I don't think you understand. My friends will make fun of me for this. It's that bad.” Here the user was so passionate about Firefox features that he resisted change. He had built social capital through his friends, and the strength of this capital was difficult to penetrate. This is, however, where the team can leverage community passion to build more social capital and work with this user, through many discussions, and perhaps get his friends involved so that discussions can continue. Through these discussions, new social capital is built, as long as the discussions continue until understanding, or satisfaction is met. Social capital builds strongly through frustrating interactions that are solved. UX participation in many ways throughout the community provides opportunities for successful interactions. Thus, with time, the community is able to understand that design proposals are explorations and not planned changes to get upset about. In addition, building social capital through interactions builds trust in UX expertise, which is an alternative to demanding respect because of expertise. Throughout the entire location bar thread, heated discussion occurred, but in the end social capital allowed people to disagree bitterly, compromise, and then move on without lasting resentment.

6.3.2 Social capital in OpenOffice.org UX

Building social capital requires building trust through successful social interactions. Because the UX team is required to participate in creating feature specifications, over time, they will build trust with other members. For example, the Notes feature included two UX members, two developers, a quality assurance person, and a documentation specialist. These people, from four different communities of practice, collaborated on the same activity to produce a feature. In the future, members of this team, because they built trust through successfully producing a feature, can ask one another for help or seek advice. For example, the Notes team could work together to fix the top Calc issue mentioned above. As such, different members of different communities of practice build social capital every time they work together successfully.

6.4 Overview of Human Development

When people engage in open-ended, highly interactive, complex problem solving in team environments over spans of time, they change. This is due to the socio-cultural aspects of learning, where a person’s thought, language, and reasoning processes are understood through social interactions with others [65].

Human development is achieved through a dialectic in which practical social activity shapes thought and at the same time regulates activity, learning, and relationships [Carroll et al. 2006]. Human development is exemplified by changes in cooperative activities, descriptions of learning, and relationship building achieved through conflict resolution. The zone of proximal development (ZPD) [64] is the space of currently possible learning achievements, attainable with the support of other actors, of information systems, or other support. Achievements within a ZPD trigger conflict resolution, enabling people to understand, act, and cooperate in more sophisticated ways. Human development favors change. Indeed, as suggested by Wenger et al., [67] communities of practice either learn and develop, or they die. Incorporating UX designers into FLOSS communities is a source of vitality to open source practices that offers new ways for the community to conceptualize and engage in a more design-centered community of practice.

6.4.1 Human development in Firefox UX

The integration of UX in Firefox includes bringing new knowledge to the discussion forums where developers interact with UX practitioners and both learn from each other. An indication of change is the promotion of the UX director to director of front-end development, user experience, and product delivery. This position provides an opportunity for human development across the Mozilla organization because the UX perspective is being perpetuated from a broader position. An example of community learning occurred in a UX blog about polishing the UI in Firefox 3. The Firefox UX team member posted several screenshots and related bugs referring to small changes in the UI that would polish the menus in Firefox 3. In the comments, five different users suggested other areas in the menu that

needed polish based on what the UX team member presented in the blog. The users took the UX expertise presented in the blog and applied it to finding similar polish problems.

6.4.2 Human development in OpenOffice.org UX

Community building by the UX team keeps both the UX community and the larger OOo community thriving. The survey results indicated that the UX team has some weaknesses to overcome, for example, finding better ways to collaborate over visual designs and encouraging more UX practitioners to participate in OOo. These weaknesses result in changes to both the tacit and explicit understanding of how the UX community can thrive. These continuous changes to the UX community, in turn, drive change in the larger OOo community, and members of each community of practice within the OOo community find new ways of engaging with each other. This continual striving to overcome challenges for the good of the community results in human development. The OOo communities continue to thrive because technical and social challenges drive change both within and outside the UX community. Therefore, as the community grows, so do its members.

6.5 UX Activity Awareness in Firefox and OpenOffice.org

Activity awareness in Firefox UX indicates where information is being understood appropriately for common ground to be reached; how communities of practice meet in a common space; where successful interactions, even if they are heated interactions, build social capital; and where learning exchanges occur through social interactions. As such, the Firefox UX and developer communities of practice are more integrated, as indicated in Figure 3. Given this integration, both communities of practice experience human development.

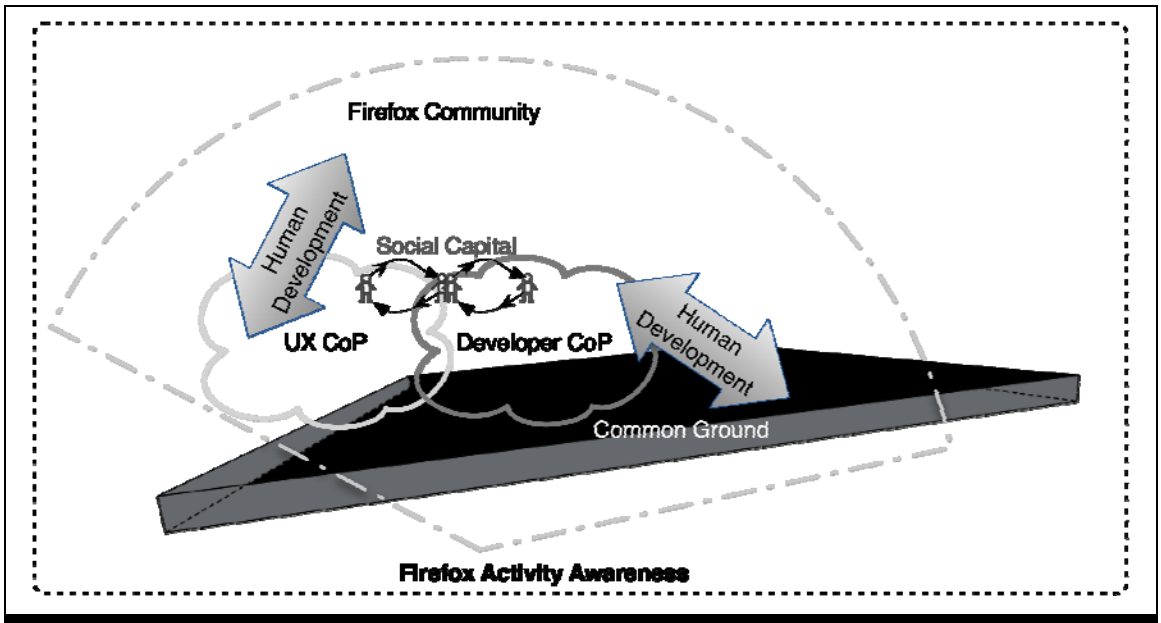

Figure 3. Activity Awareness in Firefox Open UX Design

Building a community is an important activity for OOo because of its size and complexity. The community is multinational and multidisciplinary. Therefore, awareness of the many components and projects is important for the UX team's successful integration. Because of the complexity in the community and the building awareness throughout the community, the UX and developer communities of practice have some work to do in order to become more integrated.

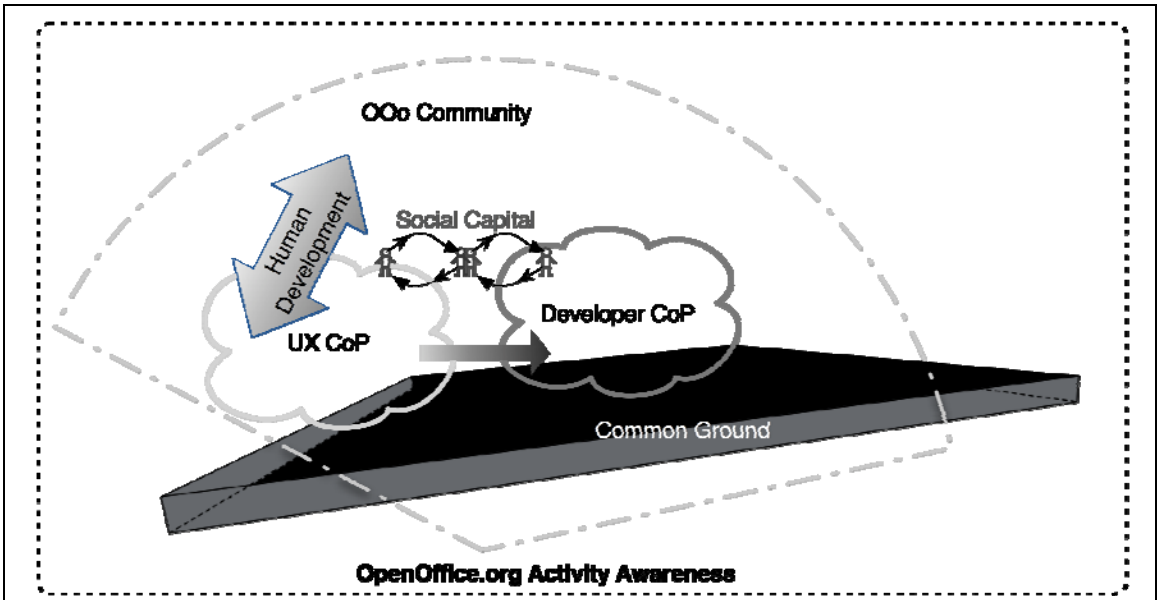

Figure 4. Activity Awareness in OpenOffice.org OpenUX Design

The activity awareness analysis uncovered differences in socio-technical structures, as seen in Figures 3 and 4. In the next section, we discuss the implications for OpenUX Design as revealed through the analysis.

7. Implications for OpenUX Design

The activity awareness analysis revealed that socio-technical structures of the FLOSS cases make integrating UX design challenging. The overarching challenges help to characterize the dynamics of open UX design. Below we discuss the different challenges faced by each project, strategies that seem to be working, and differences between the cases.

Community building in the two cases differs. Differences occur in whether emphasis is on building within the UX group, as the OOo case, or building between developers and UX contributors, as in the Firefox case. Another challenge is resolving where to integrate UX design practices. Evidence of UX design activities exists in various communication media, but the status of UX in the community determines to what extent UX design activities integrate into the larger community. Integration of UX design not only includes pushing UX design into the community, but also includes pulling UX design information from the community. The challenge is strengthening the signal despite the noise. The question is: How can open UX design activities be valued by the existing community members without the UX members continuously breaking down barriers to get their message heard? UX knowledge sharing occurs in both cases, but challenges occur with reaching common ground.

Several differences in UX activities exist between OOo and Firefox. The OOo UX community is legitimized through project status and through mandatory participation in feature development, whereas Firefox UX continually has to prove its status in the community. One explanation for status relates to size and complexity of the community. A larger, more complex community thrives better with more structure. The OOo is more complex, with several projects including different applications within the office suite, and Firefox is one application. At most, the Mozilla UX team oversees two applications: Firefox and Thunderbird mail client. The Firefox UX team is not as complex as the OOo UX team, which invites anybody to become a member. While less complexity in Firefox provides more opportunity for common ground, at the same time, it limits opportunity for change because the Firefox UX team lacks diversity, as it does not allow outside UX practitioners. This simplicity leaves less room for change and growth. Yet, additional members bring more noise to the open UX design system.

High user participation in design discussions introduces considerable noise to the system. But Firefox creates summaries on which to make decisions, and OOo posts lists of issues by top vote. The OOo UX approach resembles design by committee, whereas the Firefox UX approach resembles benevolent dictator approaches and is more efficient. Evidence of this efficiency includes Firefox UX making clearer decisions via UX design reviews in the bug tracker. Yet, the OOo UX team could strengthen the signal to noise ratio in user feedback to get a more efficient information flow. Information summarized by a UX practitioner is less noisy than a ratings list because a summary provides focused information on which to base decisions, and a ratings list merely tallies votes. The reasons users voted for changes are not known, whereas a summary provides the much needed rationale. As such, Firefox is better at crowdsourcing [35], that is, better at leveraging mass collaboration from its passionate user base. Both of these decision-making strategies, while different, offer opportunities for building trust through successful human interactions, and thus, building social capital.

Each UX community has a different strategy for UX information sharing. The Firefox UX blog disseminates UX information to the broader Firefox community, whereas the OOo UX blog asks for feedback from the OOo community. However, the OOo UX blog tries to pull information from the developers and users, and the Firefox blog tries to disseminate or push UX information to developers and users. The Firefox UX team is actively sharing information by disseminating it to the community and, therefore, actively building common ground. The OOo UX team, by pulling information into the UX community, strengthens common ground by eliciting knowledge from the broader OOo community.

From the discussion above, important characteristics of open UX design include community building, status in the community, decision making, and design information flow. See Table 2 for a summary of characteristics of open UX design. Instances of how a UX team builds community help with the entire FLOSS project in understanding the domain of open UX design. While OOo's strategy was to build a separate UXD community strong in numbers, Firefox amalgamated its UXD community across the entire community of users and developers. As a result, both of the cases found their way into the FLOSS development community of practice, albeit from different strategies. As such, community building, both within the UXD community of practice and across user contributor and developer communities of practice, is an important characteristic of integrating open UX design because it helps foster a larger, more integrated, multidisciplinary community of practice. This is akin to Fischer's notions of reflective communities [23] and communities of interest [23]. Both communities (collectively and reflectively) solve complex design problems with distributed stakeholders. While Fischer's two approaches are similar to each other, they extend ways in which we might further characterize the community-building feature of open UX design.

Table 2. Characteristics of Open UX Design in Firefox and OpenOffice.org

	Community Building	Status in Community	Decision making	Design Information Flow
OpenOffice.org	UX team expanding	Diffuse	Design by committee	Pull UX information from community
Firefox	Across UX, users & developers	Deliberate	Benevolent dictator	Push UX information into community

Open UX design must establish status in the community. This status comes from building social capital through establishing trust and merit and can be pursued deliberately through specific, consistent strategies or diffusely using existing community structures to accomplish open UX design goals. Decision making serves as a key characteristic for open UX design because it helps to establish status in the community due to its ability to offer opportunities to build trust and social capital. Hedberg and Iivari [30] have proposed an additional decision making model for UX contributions in open source projects. This model is based on the onion model [3], where the decision making resides with a few core project members. In the Hedberg and Iivari model, core UX contributors are in the decision-making loop with core software development contributors and, collectively, both groups make decisions while considering each other's perspectives. Evidence of this model was seen in the

Firefox data and partially in the OOo data. The Firefox UX director had decision-making power with core development team members, as did the UX lead in OOo. However, within the OOo UX project, we found no evidence of the UX lead exerting decision-making power in the community discussions. Instead, the OOo lead gathered information from the community in order to foster ways of designing with others. As such, decision making is an important feature of open UX design and should occur in ways that foster community building and establish status in the community.

Decision making relates to design information flow because only from knowing how the community is thinking about various designs (both proposed and implemented) and usage scenarios can well-informed design decisions be made, whether through the UX design community or a UXD leader. Design information can flow from the community to the UXD leaders or be pushed out to the community through the UXD leaders. Ideally, both of these information flow methods should be featured in open UX design to inform the entire FLOSS community. This feature of open UX design resonates with Hendry's [31] notion of design informatics, which takes an information perspective on design activities, where teams focus on optimizing information transactions. Not only is this perspective key to open UX design, but it is also useful for all design activities.

8. Implications for Activity Awareness

Revealed through the lens of activity awareness, we have presented four characteristics of open UX design, namely community building, UX status in the community, decision making, and information flow. Activity awareness was useful for analysis through the four facets -- common ground, community of practice, social capital, and human development -- because the framework included a tool for discovering challenges with different practices (community of practice), including language use and skills (common ground). Over time, project developers and designers learn how to work together to help each other understand (social capital and human development). The activity awareness framework has proven to be a useful theoretical framework for analyzing the teamwork complexities of open UX design and open source software development [4]. Because FLOSS socio-technical structures are complex, deconstructing the structures through the lenses of common ground, communities of practice, social capital, and human development illuminates characteristics important for open UX design activities and brings potential for open UX design activity awareness into open source projects.

The process of becoming aware of distributed activities open UX design and FLOSS development has been revealed. Common ground proves to be a useful protocol for analysis of communication, particularly communication across distinct knowledge bases. The integration of two distinct domains of practice shows communities of practice to be useful for seeing that the domains must come together and where their boundaries exist. The inevitable existence of conflict across the domains and the need for decision making establish social capital as a worthwhile tool for comprehending trust, mutual understanding, and shared values. Finally, the self-adjustment inherent in open source community participation coupled with conflict resolution enable people to understand, act, and cooperate in more sophisticated ways. This self-adjustment can be best understood through the lens of human development. Therefore, the activity awareness framework has been able to establish explanations for the barriers to open UX design described in Table 1. Because the literature mainly echoes the barriers, we have been able to offer explanations for challenges in open UX design by characterizing its main factors and seeing, through the four facets of activity awareness, where opportunities exist for building, supporting, and sustaining open UX design practices in FLOSS environments.

An analysis conducted using activity awareness can be costly. The intellectual costs surround learning and understanding four distinct approaches, each with its own set of constructs, and how they amalgamate into a framework for activity awareness. Furthermore, time costs include several passes over the data while analyzing through each of the four facets. Although this may be less costly than a grounded theory approach, it is more costly than employing a theoretical framework with fewer constructs to consider. The activity awareness lens, however, has other limitations pertaining to the context of open source. The main limitations exist with communities of practice and common ground. Because open UX design requires an integration of domains, communities of practice limit the scope

of how we investigate the crossover of two or more domain practices. As we have mentioned above, perhaps Fischer's notions of communities of interest or the reflective community are more useful in open source contexts, particularly when people of differing skill sets and knowledge bases must work together to produce a complex artifact, as in the case of UX design. In addition, different knowledge bases make reaching common ground challenging. Now that we have seen challenges for reaching common ground, strategies for reaching it faster and more effectively could be useful. For example, negotiation or persuasive strategies or other forms of argumentation that result in common ground are the next steps in expanding the usefulness of activity awareness. We have seen that decision making fosters building community and establishing status in the community. While the activity awareness framework revealed that social capital was important for community dynamics, it offered little guidance for understanding power structures. Thus, the next steps for understanding decision making involve a power structure analysis. We found leadership to be important for information flow. Carefully tracking and summarizing UX information is key to making it easier to consume by the community. Creating UX information systems, including technologies that support and sustain key characteristics of open UX design, is important because it helps to establish trust and merit in the community [4, 5].

9. Conclusion

While it is our goal to understand open UX design across many different types of open source projects, this paper presented data from two projects of similar type. As with all case studies, we caution generalizations to other software development organizations. The landscape of open source project communities differs greatly, as we have seen even with the two similar cases examined here. We can, however, abstract characteristics of open UX design that might be found elsewhere and, thus, help scope out a series of representations of open UX design that can be used to strategize how to best integrate UXD into software development environments, both open source and proprietary. In describing FLOSS communities of the future, we postulate that open UX design can be assimilated into the FLOSS development paradigm. While this currently presents challenges, it also presents opportunities for growth in FLOSS communities and in the whole FLOSS paradigm. This growth potential allows new categories of software development, particularly from a UX design perspective, to emerge, similar to the way FLOSS has revolutionized software development in general.

Acknowledgements

We thank the Mozilla and OpenOffice.org UX teams for sharing their practices with the world. This research was conducted by the first author as a graduate student at the Pennsylvania State University in the College of Information Science and Technology and as a computing innovation post-doctoral fellow at the University of Illinois Graduate School of Library and Information Science. This work was partially supported by the open source software lab at Microsoft and partially supported by the National Science Foundation under Grant #0937060 to the Computing Research Association for the CI Fellows.

References

[1] Aaltonen, T. and Jokinen, J. "Influence in the Linux Kernel Community" *Open Source Development, Adoption and Innovation*, 2007.

[2] Andreasen, M. S., Nielson, H. V., Schroder, S. O. and Stage, J. Usability in Open Source Software Development: Opinions and Practice. *Information Technology and Control*, 35, 3 2006, 303-312.

[3] Antikainen, M., Aaltonen, T., Vaisanen, J., Feller, J., Fitzgerald, B., Scacchi, W. and Silitti, A. "The role of trust in OSS communities -- A Case Linux Kernel community" *Open Source Development, Adoption and Innovation* Springer, New York, 2007.

[4] Bach, P. M. *Supporting the User Experience in Free/Libre/Open Source Software Development*. Dissertation, The Pennsylvania State University, University Park, 2009.

[5] Bach, P. M., DeLine, R. and Carroll, J. M. Designers Wanted: Participation and the User Experience in Open Source Software Development. In *Proceedings of the CHI 09* (Boston, MA USA, 2009). ACM.

[6] Bach, P. M., Jiang, H., and Carroll, J. M. *Sharing usability information in interactive system development*. ACM, New York, NY, City, 2008.

[7] Basset, T. and Stefan, K. "Coordination and Social Structures in an Open Source Project: VideoLAN" *Free/Open Source Software Development* Idea Group Publishing, Hershey, PA, 2004.

[8] Bertelson, O. W. and Bodker, S. "Activity Theory" *HCI Models, Theories, and Frameworks*. Ed. J. M. Carroll Morgan Kaufmann, Los Altos, CA, 2003.

[9] Bodker, S. *Through the Interface: A Human Activity Approach to User Interface Design*. Lawrence, Erlbaum Associates, City, 1991.

[10] Boivie, I., Gulliksen, J. and Goransson, B. The lonesome cowboy: A study of the usability designer role in systems development. *Interacting with Computers*, 18, 4 2006, 601-634.

[11] Carroll, J. M., Rosson, M. B., Convertino, G. and Ganoe, C. Awareness and teamwork in computer-supported collaborations. *Interacting with Computers*, 18 2006, 21-46.

[12] Carroll, J. M., Rosson, M. B., Farooq, U. and Xiao, L. Beyond being aware. *Information and Organization*, 19, 3 2009, 162-185.

[13] Cataldo, M., Wagstrom, P. A., Herbsleb, J. D. and Carley, K. M. *Identification of coordination requirements: implications for the Design of collaboration and awareness tools*. ACM, City, 2006.

[14] Clark, H., Brennan, S. E., Resnick, L. B., Levine, J. M. and Teasley, S. D. "Grounding in communication" *Perspectives on socially shared cognition* American Psychological Association, Washington, DC, 1991.

[15] Clark, H. *Using Language*. Cambridge University Press, New York, 1996.

[16] Coleman, J. S. Social capital in the creation of human capital. *American Journal of Sociology*, 94 1988, 95-120.

[17] Crowston, K., Annabi, H., Howison, J. and Masango, C. *Effective Work Practices for Software Engineering: Free/Libre Open Source Software Development*. City, 2004.

[18] Crowston, K. and Howison, J. The Social Structure of Free and Open Source Software Development. *First Monday*, 10, 2 2005.

[19] Daft, R. L. and Lengel, R. H. Organizational Information Requirements, Media Richness and Structural Design. *Management Science*, 32, 5 1986, 554-571.

[20] Ducheneaut, N. Socialization in an Open Source Software Community: A Socio-Technical Analysis. *Computer Supported Cooperative Work*, 14, 4 2005, 323-368.

[21] Engestrom, Y. *Learning, Working, and Imagining: Twelve Studies in Activity Theory*. Orienta-Konsultit Oy, Helsinki, 1990.

[22] Feller, J. and Fitzgerald, B. *Understanding Open Source Software Development*. Addision-Wesley, London, 2002.

[23] Fischer, G. *Communities of Interest: Learning through the Interaction of Multiple Knowledge Systems*. City, 2001.

[24] Fitzgerald, B. The Transformation of Open Source Software. *MIS Quarterly*, 30, 3 2006, 587-559.

[25] Fitzgerald, B. The Transformation of Open Source Software. *MIS Quarterly*, 30, 3 2006, 587-559.

[26] Gao, Y. and Madey, G. "Network Analysis of the SourceForge.net Community" *Open Source Development, Adoption and Innovation*, 2007.

[27] Ghosh, R. A., Feller, J., Fitzgerald, B., Hissam, S. A. and Lakhani, K. R. "Understanding Free Software Developers: Findings from the FLOSS study" *Perspectives on Free and Open Source Software* MIT Press, Cambridge, 2005.

[28] Hassenzahl, M. and Tractinsky, N. User expereince - a research agenda. *Behavior and Information Technology*, 25, 2 2006, 91-97.

[29] Hedberg, H., Iivari, N., Rajanen, M. and Hajumaa, L. *Assuring Quality and Usability in Open Source Software Development*. IEEE, City, 2007.

[30] Hedberg, H. and Iivari, N. *Integrating HCI Specialists into Open Source Software Development Projects*. Springer, City, 2009.

[31] Hendry, D. G. *Design Informatics – Information Needs in Design 2007 Sept. 30, 2009* http://swiki.cs.colorado.edu:3232/CHI07Design/uploads/5/hendry.pdf

[32] Herbsleb, J. and Mockus, A. An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering*, 29, 6 2003, 481-494.

[33] Herbsleb, J. D. *Global Software Engineering: The Future of Socio-technical Coordination*. IEEE Computer Society, City.

[34] Hine, C. *Virtual Ethnography*. Sage, Thousand Oaks, CA, 2000.

[35] Howe, J. *The Rise of Crowdsourcing*. Conde Nast, City, 2006.

[36] Howison, J., Inoue, K., Crowston, K., Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M. and Succi, G. "Social dynamics of free and open source team communications" *IFIP International Federation for Information Processing* Springer, Boston, 2006.

[37] Howison, J. *Alone Together: A socio-technical theory of motivation, coordination and collaboration technologies in organizing for free and open source software development*. Dissertation/Thesis, Syracuse University, 2009.

[38] Iivari, N. 'Representing the User' in software development--a cultural analsyis of usability work in the product development context. *Interacting with Computers*, 18 2006, 635-664.

[39] Koch, S. "Exploring the Effects of Coordination and Communication Tools on the Efficiency of Open Source Projects using Data Envelopment Analysis" *Open Source Development, Adoption and Innovation*, 2007.

[40] Kuniavsky, M. *Observing the User Experience: A Practitioner's Guide to User Research*. Morgan Kaufmann, San Francisco, 2003.

[41] Lave, J. and Wenger, E. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, New York, 1991.

[42] Mockus, A., Fielding, R. T. and Herbsleb, J. Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11, 3 2002, 309-346.

[43] Moon, J. Y. and Sproull, L. Essence of distributed work: the case of Linux kernel. *First Monday*, 5, 11 2000.

[44] Mueller-Prove, M. User Experience for OpenOffice.org. *Interfaces*, 71, Summer 2007, 8-9.

[45] Nardi, B. A. *Context and consciousness : activity theory and human-computer interaction*. MIT Press, Cambridge, Mass., 1996.

[46] Nichols, D. M. and Twidale, M. B. *The Usability of Open Source Software 2003* Accessed from http://firstmonday.org/issues/issue8_1/nichols/index.html on Jan. 15, 2006.

[47] Nichols, D. M., Twidale, M. B. and McKay, D. *Participatory Usability: supporting proactive users*. ACM, City, 2003.

[48] Nichols, D. M. and Twidale, M. B. Usability processes in open source projects. *Software Process: Improvement and Practice*, 11, 2 2006, 149-162.

[49] Raymond, E. S. The cathedral and the bazaar. *First Monday*, 3, 3 1998.

[50] Roberts, J., Hann, I. H., Slaughter, S., Damiani, E., Fitzgerald, B., Scacchi, W. and Succi, G. "Communication Networks in an Open Source Software Projects" *IFIP International Federation for Information Processing* Springer, Boston, 2006.

[51] Rosson, M. B. and Carroll, J. M. *Usability Engineering: Scenario-Based Development of Human Computer Interaction*. Morgan Kaufmann, San Francisco, 2001.

[52] Sack, W., Detienne, F. o., Ducheneaut, N., Burkhardt, J.-M., Mahendran, D. and Barcellini, F. A Methodological Framework for Socio-Cognitive Analyses of Collaborative Design of Open Source Software. *Computer Supported Cooperative Work (CSCW)*, 15, 2 2006, 229-250.

[53] Scacchi, W. and Bainbridge, W. S. "Socio-Technical Design" *The Encylcopedia of Human-Computer Interaction* Berkshire Publishing Group, Great Barrington, 2004.

[54] Scacchi, W. "Socio-Technical Interaction Networks in Free/Open Source Software Development Processes" *Software Process Modelling*. Ed. S. T. Acuna and N. Juristo Springer, New York, 2005.

[55] Scacchi, W., Acuna, S. T. and Juristo, N. "Socio-Technical Interaction Networks in Free/Open Source Software Development Processes" *Software Process Modelling* Springer, New York, 2005.

[56] Scacchi, W. and Zelkowitz, M. "Free/Open Source Software Development: Recent Research Results and Methods" *Advances in Computers* Academic Press, San Diego, 2007.

[57] Schmidt, K. The problem with 'awareness'. *Computer Supported Cooperative Work*, 11 2002, 285–298.

[58] Seffah, A., Gulliksen, J. and Desmarais, M. *Human Centered Software Engineering*. Springer, City, 2005.

[59] Sharp, H., Rogers, Y. and Preece, J. *Interaction Design: Beyond Human-Computer Interaction* Wiley, Hoboken, 2007.

[60] Studer, M. "Community Structure, Individual Participation and the Social Construction of Merit" *Open Source Development, Adoption and Innovation*, 2007.

[61] Trist, E. and Murray, H. *The Social Engagement of Social Science: A Tavistock Anthology, The Socio-technical Perspective*. University of Pennsylvania Press, Philadelphia, PA, 1993.

[62] Twidale, M. B. and Nichols, D. M. Exploring Usability Discussions in Open Source Development. In *Proceedings of the 38th annual Hawaii International Conference on System Sciences (HCICS, 05)* (Hawaii, 2005, 2005). IEEE Computer Press Society.

[63] Viorres, N., Xenofon, P., Stavrakis, M., Vlachogiannis, E., Koutsabasis, P. and Darzentas, J. "Major HCI Challenges for Open Source Software Adoption and Development" *Online Communities and Social Computing*, 2007.

[64] Vygotsky, L. S. *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, City, 1978.

[65] Vygotsky, L. S. and Kozulin, A. *Thought and Language*. MIT Press, Cambridge, MA, 1987.

[66] Wenger, E. *Communities of Practice: Learning, Meaning, and Identity*. Cambridge University Press, Cambridge, 1998.

[67] Wenger, E., McDermott, R. and Snyder, W. M. *Cultivating Communities of Practice*. Harvard Business School Press, Boston, 2002.

[68] Yamauchi, Y., Yokozawa, M., Shinohara, T. and Ishida, T. Collaboration with Lean Media: How Open-Source Software Succeeds. In *Proceedings of the CSCW'00* (Philadelphia, PA USA, 2000). ACM Press.

About the Authors

John M. CARROLL is Edward Frymoyer Chair Professor of Information Sciences and Technology at the Pennsylvania State University. His research interests include community informatics and methods and theory in human-computer interaction, particularly as applied to networking tools for collaborative learning and problem solving, and the design of interactive information systems. He has written or edited 14 books, including Making Use (MIT Press, 2000), HCI in the New Millennium (Addison-Wesley, 2001), Usability Engineering (Morgan-Kaufmann, 2002, with M.B. Rosson) and HCI Models, Theories, and Frameworks (Morgan-Kaufmann, 2003). He serves on 9 editorial boards for journals, handbooks, and series; he is a member of the US National Research Council's Committee on Human Factors and Editor-in-Chief of the ACM Transactions on Computer-Human Interactions. He received the Rigo Award and the CHI Lifetime Achievement Award from ACM, the Silver Core Award from IFIP, the Alfred N. Goldsmith Award from IEEE, and is an ACM Fellow.

Paula BACH is a UX researcher at Microsoft in the developer division where she studies how software developers work. Previously she was a postdoctoral research associate at The University of Illinois Graduate School of Library and Information Science (GSLIS) working with Mike Twidale and funded through the NSF Computing Innovation Fellowship program. Bach completed her thesis, “Supporting the User Experience in Free/Libre/Open Source Software (FLOSS) Development” working in the Computer Supported Collaborative Learning Lab and Penn State Center for HCI under the direction of Jack Carroll, Edward M. Frymoyer professor of Information Sciences and Technology. She is still working to continue and extend her research on ways in which sociotechnical solutions can foster participation from HCI professionals and HCI interested users as well as investigating other ways to make FLOSS more usable.