Design Information Sharing Across Multiple Knowledge Systems in a FLOSS Community

Paula M. Bach
College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802
pbach@ist.psu.edu

ABSTRACT
This paper explores support for design information sharing between the distinct knowledge systems and skill sets of interactive system designers and developers. The paper focuses on the challenges of sharing information among groups of designers, developers, and users with multiple knowledge systems in the context of free/libre/open source software (FLOSS) communities. Bringing design to FLOSS communities introduces new knowledge into a solitary community of practice, and discussion ensues about how exploiting the 'symmetry of ignorance' can enhance information sharing through design in CodePlex, an open source project hosting community website. Finally, design mockups illustrate how CodePlex serves as a boundary object supporting design information sharing across distinct knowledge systems.

Categories and Subject Descriptors
H.5.3 [**Group and Organization Interfaces**]: Collaborative computing.

Keywords
User experience design, design, software development, software engineering, communities of practice, boundary objects, open source software, FLOSS, information sharing.

1. INTRODUCTION
The purpose of this paper is to explore how to support design information sharing between the different perspectives espoused by distinct knowledge systems and skill sets of interactive system designers and developers. I explore this purpose in the context of a free/libre/open source (FLOSS) community to investigate user experience design at the level of information sharing because the different knowledge systems of designers and developers pose a challenge for sharing design information. The exploration begins by presenting literature on design information sharing and the social-psychological effects of information sharing. It continues by exploring challenges with bringing design to FLOSS environments where developer knowledge systems prevail and demonstrating how information sharing across design and developer knowledge systems in one FLOSS community can be supported through design mockups that, if implemented, serve as proposed boundary objects.

iConference'09, February 8–11, 2009, Chapel Hill, NC, USA.

1.1 Design Information Sharing
In general, design information is important not only for making user experience design decisions but also for making decisions about the software's architecture. Designers are responsible for researching, designing, and evaluating interactive systems taking special care to create a desirable user experience. Software developers are responsible for specifying and implementing the system taking special care that the software is free of errors and works according to specification.

Decisions made within design teams affect the technical aspects of implementing the software and conversely decisions made within the development teams affect the user experience. Therefore information from both disciplines is essential for making the right decisions about user experience design. Because design information is important, effective information sharing practices and tools are essential.

Sharing design information across interactive system development teams is essential for making design decisions. Personal connections and interactions that transcend organizational boundaries, such as the separation between development teams and design teams, support the flow of information [15]. The idea is to package information so that many different people can use it. For example, developers need information to understand the designs they are coding [13]. Such design information needs that developers may experience include answers to the following questions: What is the program supposed to do? Is this problem worth fixing? What are the implications for this change? These needs connect to design information because changes could affect the user experience. In addition, software design teams have collective information needs including identifying information needs, formulating information queries, retrieving relevant information, and communicating about information needs [13].

While the above research indicates challenges already uncovered with sharing information in an interactive system design and development context, other research investigates the social-psychological effects of information sharing. The following discussion presents social-psychological factors prevalent in information sharing with possible examples from the interactive system context. Because psychological costs exist for asking about information, individuals follow the law of least effort [3]. For example, if a designer is concerned about making a deadline she might ask a developer how technical a redesign is. On one hand, she could wait until the team meeting or drop by the developer's office to discuss the implications of the design rework. The designer might be concerned however, that she does not understand the technical details of the fix and the developer may perceive incompetence. This is a psychological cost of information sharing. Contrary to psychological costs of

information sharing is the benefit of shared experiences. Shared experiences in information sharing benefit teams by several mechanisms. One such mechanism, the common knowledge effect, states that an item of information will have more influence on the judgment of a group when it is shared [7]. For example, when developers and designers share an item of information about a design issue, that item is deemed more important when it is shared compared to if was not shared among team members. Another mechanism, shared context, proposes that developing a shared interpretive context helps groups lacking substantial history [23]. Therefore, a shared context, such as a common development methodology or motivation to solve a particular design issue should provide a better information-sharing context for interdisciplinary teams with varying backgrounds. A final mechanism for beneficial information sharing is the need for dense social networks [17]. These networks promote frequent communication among team members about work context and arising situations. These social aspects of information sharing can impact the quality of information sharing and account for breakdowns or successes.

We have seen that information sharing in general is complex as several factors can affect effectiveness among groups with different knowledge systems. In the context of artifact creation, and in this case software design and development, multiple perspectives are necessary to explore design possibilities. Yet, sharing information among groups with multiple perspectives is challenging. Thus, in design, we have the problem of sharing information across multiple perspectives. On one hand, multiple perspectives are good for design exploration and innovation, yet, on the other hand, they hinder the information sharing that is needed for design exploration and innovation.

We explore this design and information sharing contradiction, by investigating design in a particular context: free/libre/ open source software development communities. This context is interesting to explore because upfront design, especially designing for the user experience, in open source communities is rare, [11, 12] and this brings forth the opportunity to understand information sharing among multiple perspectives through supporting design activities in an open community.

2. BRINGING DESIGN TO FLOSS

In FLOSS communities developers produce software that is available for download and use by anybody. 'Anybody' is somewhat misleading because, for the most part, FLOSS is not easily used by those other than developers or tech-savvy members of the open source community who are willing and able to tinker with the software to get it to work. FLOSS project members are always happy to have their software used by 'anyone'. Many projects, however, have little consideration for the user experience, including user-centered design lead by a designer. FLOSS projects like Mozilla Firefox (web browser), OpenOffice.org (office suite), and KDE (desktop environment for Linux operating system) have deliberate user experience strategies. Yet, projects like those hosted on open source project hosting websites like Google Code, SourceForge, and CodePlex lack deliberate strategies for improving the user experience [1]. Contributors to FLOSS projects submit code, leave feedback, or help other users. Contributors take roles as either a core developer who makes decisions and plans the direction of the software, a user/developer who uses the software and submits patches when something goes wrong, or a user who uses the software and participates in discussion forums and email lists and lets the core developers know when they don't like new changes or features in the software, for example [2].

Because FLOSS communities foster participation from users, the environment suits user-centered design and even participatory design. Yet, designers must harness the valuable information from users and translate it into designs that improve the user experience. In FLOSS, design means designing with the community of users engaged in the direction of the software. But this design happens only after users have tried the software, if at all, and rarely with designers. In the larger projects, like Firefox, user experience designers guide the planning stages with input from the community. Yet the smaller projects have little or no input from designers [1].

In the next sections we present an analytical argument for bringing design to FLOSS and argue that it can happen in three ways. The claims made are based on logic and conjecture because few studies have confirmed the specific challenges encountered while bringing design to FLOSS. The discussion, however, illustrates challenges for design information sharing that later are addressed through the design mockups. First, core developers can gain some design skills and rethink processes and workflows. In this case, the developer initiates change. Second, designers can participate, either by being paid, or by volunteering. In this case, either the project (by agreeing to pay a designer) or designers initiate the change (by volunteering). Third, the FLOSS community socio-technical infrastructure can support design activities so that such practices are a cultural norm in the community [1]. In this case, the FLOSS community initiates the change. In the next sections we explore each of these three means of bringing to design to FLOSS communities.

2.1 Developers Engaging in the User Experience

In the first case, the likelihood of project coordinators rethinking their strategy to include designing for the user experience is low. FLOSS developers are motivated by learning and fun [9, 22]. Furthermore, a well-known fact in the FLOSS world is that developers want people to download and use their software and that a high number of downloads is one measure of success. Thus we argue that while project coordinators and developers may not be thinking about the user experience in terms of how to improve their project's usability or look and feel, for example, but that they care about the user's experience because they want people to download and use their software. We make the assumption that because coordinators and developers care about the user's experience, they might be motivated to learn about how to improve the user's experience by focusing more on design and perhaps even have fun doing so. By this reasoning, a possibility exists that under the right circumstances, project coordinators and developers will engage in improving the user's experience of their projects by focusing on design activities.

2.2 Designers Engaging in FLOSS

In the second case, paying designers to work on FLOSS projects has been successful for larger projects within a single community, but has little success for projects hosted on a website, mainly because these projects tend to be smaller in scope. For example, the SourceForge community FLOSS project hosting website supports a marketplace where professionals can offer services including usability testing and UI design. But interactions

between designers and projects are not transparent. Furthermore, because projects hosted on community FLOSS sites tend to be smaller and without a business model, paying designers does not make sense in many cases. If paying designers to participate in FLOSS is not feasible for projects hosted on FLOSS community project hosting websites, then how can designers be motivated to participate? Volunteer designer participation in FLOSS has been much slower than developer participation. This slow uptake relates to FLOSS developer culture.

2.2.1 Clashing Cultures

The FLOSS developer culture characterizes attitudes and behaviors consistent with developing knowledge and skills. The theoretical concept of communities of practice [20] includes components of a framework that describe FLOSS developer culture. Such components are useful for talking about FLOSS communities of practice. Communities of practice are "groups of people who share a concern or a passion for something they do and learn how to do it better as they interact regularly" [21]. The community of practice framework posits four components that describe learning in terms of practice, community, identity, and meaning. In the framework, *practice* encompasses learning through doing, or active learning through participation. In a *community*, members understand and enact social configurations. Members gain *identity* by learning how to become part of the community. When members realize changes and reflect on experiences related to learning in the community they create *meaning*. In FLOSS culture, developers gain knowledge and skill by participating in a project. Such participation includes submitting or committing code or contributing to discussion forums. FLOSS project communities have different social configurations and members must understand the different hierarchies and procedures for participation [10]. For example, if a developer who uses the software wants to submit code for a new feature, he must find where to start, the lifecycle of a work item, the workflow, or examples of code. If the developer figures out how to negotiate the community and have his feature accepted, then he gains the trust of other project developers and coordinators. If the coordinators and core developers like the feature idea and the accompanying code, they'll ask for more contributions, then the developer becomes a valuable member of the community. FLOSS communities are based on trust and merit [6, 19]. Once a developer has code committed, she identifies with the community because she has been accepted. FLOSS developers, in the spirit of openness, often share their challenges and solutions while solving a coding problem. They share experiences in a weblog or in FLOSS project forums so other developers can learn from their experiences and therefore gain meaning through reflection and sharing experiences. FLOSS culture is characterized by practices, community, identity, and meaning that is incommensurate with designer culture. In short, the FLOSS developer communities of practice do not integrate well with designer communities of practice.

User experience designers advocate the user by researching possible use scenarios and ensuring designs meet usability specifications, and overall optimizing the software so that the user has a positive experience while using the software. In FLOSS, design communities of practice are not well established [11]. Design communities exist around the larger projects such as OpenOffice.org and Mozilla, but are absent among projects hosted on FLOSS community websites. These larger FLOSS UX communities have had to appropriate FLOSS developer communities of practice by using bug trackers and email lists for design [12]. The challenge resides in how design happens in FLOSS development and how to share design information among users, developers, and designers. This is a puzzling situation because FLOSS design communities of practice and FLOSS development communities of practice are different. This is an obvious point, but we must understand where the developer[1] and designer communities of practice clash.

2.2.2 Bridging Cultures

On a basic level user experience design is about people and software development is about code. In FLOSS, code is currency. Although FLOSS development and design are about creating FLOSS software, their priorities and goals differ. FLOSS communities of practice are more widely established than FLOSS design communities of practice. Furthermore, technological infrastructure exists for FLOSS development but not for design. This analysis reveals limitations to integrating two communities of practice. Active learning through practice reveals three challenges.

First, developers interested in different aspects of design including the user experience and usability have no basis for proving their merit and other developers have no way of assessing design skills and no way of understanding what to trust about design because it is outside of their domain of expertise. This describes the problem of merit and trust. Second, designers, paid or volunteer, generally have limited coding skills and are likely not highly skilled developers. Furthermore, design work does not equal developer work. The designers assess user needs, create designs, and evaluate how well the software meets users' needs, whereas the developers solve step-by-step technical problems to get software to work. Writing code is not the main activity for designers. Thus the developer community has no basis for designer merit in a FLOSS environment. This describes the problem of a chasm between work activities. Third, traditional design methodologies do not integrate into the open source distributed environment [8, 16] and designers and developers use different tools [11]. This describes the problem of incommensurable methodologies and tools. These challenges of merit and trust, tool use, and methodological incompatibility highlight why information sharing is challenging in an interactive system design and development context. As such, focusing on the characteristics of the community helps illuminate where solutions to the information sharing challenges may emerge.

What kind of a community could exist, where merit and trust serve as social configurations that must be navigated for learning in the community? Both designers and developers would have difficulty negotiating what merit and trust encapsulate. Sorting out challenges with learning in the community and establishing practices commensurate with both FLOSS design and development foster an environment where meaning and identity emerge. Identity is synonymous with a FLOSS culture that supports learning about both design and development. Problems of merit and trust, chasm between work activities, and incommensurable methodologies and tools illuminate the different knowledge systems at work between design and development communities of practice and as such provide key areas where

[1] The developer communities include users, so FLOSS communities include developers and users.

design information sharing is needed. These key areas open up opportunities for supporting design information sharing.

2.3 Community Support for Design Information Sharing

How can a FLOSS community support design information sharing of both designer and developer knowledge systems? Fischer [4] argues that because communities of practice work in domains where sustaining engagement and collaboration lead to boundaries, boundaries are set up between participants and non-participants. He suggests then that communities of practice are homogeneous and proposes heterogeneous communities of interest (CoIs). Communities of interest learn to communicate and learn from other members in the community despite having different perspectives and knowledge systems [4]. Fischer's perspective extends Rittel's notion of *symmetry of ignorance* [14] to characterize communities of interest where a shared interest exists in the community to frame and resolve a design problem. Rittel's notion of *symmetry of ignorance* posits that members involved in solving design problems exhibit both expertise and ignorance and Fischer [5] argues such expertise and ignorance *across* different knowledge systems can be used to foster creativity and innovation in collaborative design communities—which he calls communities of interest.

We propose reframing the problem of different knowledge systems in FLOSS communities as an opportunity for creativity and innovation and looking at the information-sharing problem in bringing design to FLOSS as a community of interest. Such a reframing establishes a need for socio-technical support for FLOSS communities of interest. Such support functions as a *boundary object* [18] where different knowledge systems interact via a shared reference that is meaningful within both systems [4]. Through this shared reference, design information sharing becomes possible. With additional support for development *and* design, a FLOSS community website functions as a boundary object. As such the community website has meanings across boundaries of design and development knowledge systems and fosters support for design information sharing.

3. ESTABLISHING CODEPLEX AS A COMMUNITY OF INTEREST

CodePlex is an open source project hosting community website. The CodePlex community serves as a good test bed for bringing design to FLOSS and investigating how to support design information sharing because the site is still growing (launched in 2006). As such, our research team has been working with the CodePlex community to design support for a community of interest that includes both developers and designers. Table 1 shows statistics for CodePlex in July 2008.

Page views	6,761,755
Visits	1,684,502
Unique visits	1,006,716
Registered users	92,172
Total projects	5567
New projects	385

Table 1: CodePlex July 2008 Statistics

The CodePlex community website is built and maintained by Microsoft and hosts open source projects of all types and allows projects to choose many different open source licenses. The research project included thirteen weeks of on site interviews and observations studying how the CodePlex development team supports the community. Researchers worked with the development team to produce design mockups intended to support design knowledge systems and information sharing, in addition to the support for developer knowledge systems already in place.

CodePlex supports development activities with socio-technical features. Technical features include code repository, issue (bug) tracker, project definition space (a wiki), and a release and download area. Social features include discussions (in forum and at an individual issue), roles for project members, votes for changes, and ratings for releases, among other community support features. The CodePlex team works closely with the developers and users to support the needs of the community. For example, the community suggested that a rating and voting systems would be helpful and the CodePlex team developed such support features. In the next section mockups present a first order investigation into exploring support for design in a FLOSS environment. Design rationale for the mockups occurs as information sharing across the design and development knowledge systems. As such, for CodePlex to function as a boundary object, it has to allow design and development knowledge systems to interact by providing a shared reference that is meaningful within both systems.

3.1 Opportunities for Information Sharing Across Knowledge Systems

The design goals were to find places where design and development knowledge systems could interact and ensure opportunities for information sharing across the knowledge systems so that the information was meaningful for both systems. Features for CodePlex support bringing design to FLOSS. Other FLOSS environments that have already integrated design knowledge systems do so without adding new features (e.g. Firefox), while others do so by adding separate web space in the community and email distribution lists (e.g. openoffice.org). The strategy was to incorporate the new design space at the same level as development space while allowing for both developers and designers to learn about each environment, and thus experience each other's knowledge system. In meeting the goal of providing interaction space for design and development knowledge systems and ensuring opportunities for information sharing across the knowledge, we address the three challenges for integrating communities of practice as design solutions. Such challenges included opportunities for trust and merit, interaction of design and development work activities, and incorporation of design tools and methods. The following design mockups serve as first order solutions for the challenges.

3.2 Design Workspace

The design workspace augments other sections of the project area. On CodePlex, each open source project includes sections, divided and labeled as tabs, where developers coordinate and collaborate on project activities. Sections include areas for source code, discussions, team members, releases, and so on. Adding a design workspace tab allows for a distinct area where designers can work on activities using methods and tools that are familiar to them. At the same time, developers can try out design activities. For example, a scaffolded approach to learning how to create scenarios and personas helps developers interact with designer

knowledge systems because creating and working with scenarios and personas are common knowledge for most designers. See figure 2 for various available design activities.

3.3 Space for Design Interactions

Accessible from the design workspace, the space for design allows designers, developers, and users to upload images as prototypes for various design ideas, and discuss and comment on the design ideas. Each design idea can go through several iterations and when a design idea is ready for coding it can be submitted and linked directly to a work item in the issue tracker. This workflow presents an opportunity for the interaction of developers and designers. These interactions, where contributors are sharing work activities, provide favorable conditions for establishing trust and merit because a developer can observe designs submitted by a designer and interact in discussion thereby gaining experience understanding of the design work process and knowledge system. At the same time, designers can trace their designs to a work item in the issue tracker and conduct design reviews as the developer is coding a design mockup. See figure 1 for an example of the design space.

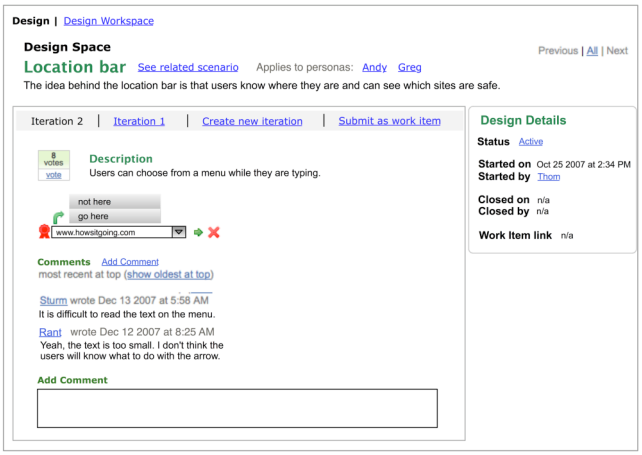

Figure 1: Design space where contributors interact.

4. DISCUSSION

As a boundary object for the interaction of design and development knowledge systems, CodePlex currently does not support design activities. Bringing design to CodePlex involves keeping knowledge systems separate but offering opportunities for interaction, and as such exploit the *symmetry of ignorance* experienced by community members. Because designers, developers, and users, can interact in the new design space, challenges of merit and trust, work activities, and tools and methods, would be mitigated and at the same time, CodePlex projects have the opportunity to be more innovative with the introduction of multiple knowledge systems.

With the introduction of a new knowledge system the information-sharing trail is important because it serves as source of learning through creation and discussion. Discussions across knowledge systems allow opportunities for connections. For example, the interaction of translating designs to code involves multiple discussions among the community members with opportunities for sharing new information based on one's knowledge and expertise. As such, the information trail is mapped through both knowledge systems. Information sharing in the new environment occurs in a shared context and offers opportunities for the common knowledge effect to influence a project. For example, a design idea that has many iterations and comments serves as a shared context. When an often-shared (many contributors participate in discussion and uploading new iterations of a design idea) design idea is translated to code and entered into the issue tracker, it continues to create an information trail. Furthermore, the psychological costs are alleviated to some extent because the participation is virtual and potential contributors can lurk on the list and learn the rules for participation and information sharing, and when they feel comfortable enough to participate, their psychological risk is reduced. Therefore, because information sharing is virtual, contributors can contribute in a limited way (e.g. a simple comment on a design idea), following the law of least effort, until they are comfortable and contribute at another level, perhaps by uploading their own design ideas. Information sharing across the different environments representing the different knowledge systems allows for the right balance between expertise and ignorance. As such, the new design support features refine CodePlex as a boundary object, so that it can allow knowledge systems to interact through information sharing.

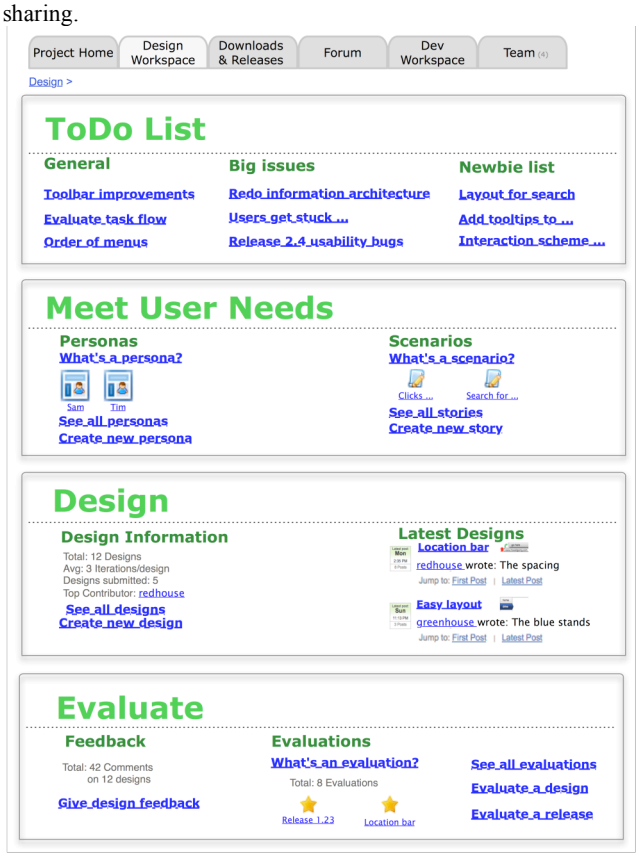

Figure 2: Design workspace provides opportunities for interaction between knowledge systems.

5. CONCLUSION

This paper has explored support for design information sharing between the different perspectives espoused by the distinct knowledge systems and skill sets of interactive system designers and developers. The exploration revealed that refining CodePlex as a boundary object that supports information sharing across knowledge systems fosters innovation that has the potential to enhance the overall user experience that could result in better

software for users of all levels. As such, this paper reports on the first order design phase as rationale for bringing design to CodePlex by investigating design information sharing across different knowledge systems. We continue to work with the CodePlex team to understand the multiple ways CodePlex can be enhanced by bringing design to the open source projects hosted on the community website. Future research seeks to assess and articulate specific mechanisms for how design information sharing bridges multiple knowledge systems via boundary objects designed and implemented in CodePlex.

6. ACKNOWLEDGMENTS

The open source software lab at Microsoft sponsors this research and the CodePlex study is in partnership with Microsoft Research. The author thanks the reviewers for their helpful comments.

7. REFERENCES

[1] 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 Conference on Computer-Human Interaction* (Boston, MA, 2009). ACM.

[2] Crowston, K., Annabi, H., Howison, J. and Masango, C. Effective Work Practices for Software Engineering: Free/Libre Open Source Software Development. In *Proceedings of the Workshop on Interdisciplinary Software Engineering 2004, SIGSOFT 2004* (Newport Beach, CA, 2004).

[3] Dewhirst, D. H. Influence of Perceived Information-Sharing Norms on Communication Channel Utilization. *The Academy of Management Journal*, 14, 3 1971, 305-315.

[4] Fischer, G. Communities of Interest: Learning through the Interaction of Multiple Knowledge Systems In *Proceedings of the Information Systems Research Seminar in Scandinavia* (Bergen, Norway, August 11-14, 2001).

[5] Fischer, G. Social Creativity: Turning Barriers into Opportunities for Collaborative Design. In *Proceedings of the Participatory Design Conference* (Toronto, Ontario, Canada, July 27 -31, 2004). ACM.

[6] Gallivan, M. J. Striking a Balance Between Trust and Control in a Virtual Organization: a content analysis of open source case studies. *Information Systems Journal*, 2001, 11 2001, 277-304.

[7] Gigone, D. and Hastie, R. The Common Knowledge Effect: Information Sharing and Group Judgement. *Journal of Personality and Social Psychology*, 65, 5 1993, 959-974.

[8] Hedberg, H., Iivari, N., Rajanen, M. and Hajumaa, L. Assuring Quality and Usability in Open Source Software Development. In *Proceedings of the Proceedings of the First international Workshop on Emerging Trends in FLOSS Research and Development* (ICSE '07, May 20 - 26, 2007). IEEE.

[9] Lakhani, K. R. and Wolf, R. G. Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. *Perspectives on Free and Open Source Software*. Ed. J. Feller, B. Fitzgerald, S. A. Hissam and K. R. Lakhani MIT Press, Cambridge, 2005.

[10] Muffatto, M. *Open Source A Multidisciplinary Approach*. Imperial College Press, London, 2006.

[11] Nichols, D. M. and Twidale, M. B. *The Usability of Open Source Software 2003* Accessed from http://firstmonday.org/issues/issue8_1/nichols on 14 July, 2006.

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

[13] Poltrock, S., Grudin, J., Dumais, S., Fidel, R., Bruce, H. and Pejtersen, A. M. Information Seeking and Sharing in Design Teams. In *Proceedings of the Group '03* (Sanibel Island, FL, 2003). ACM.

[14] Rittel, H. Second-Generation Design Methods. *Developments in Design Methodology*. Ed. N. Cross. John Wiley & Sons, New York, 1984.

[15] Salvador, T. and Bly, S. Supporting the flow of information through constellations of interaction. In *Proceedings of the European Computer Supported Cooperative Work* (Lancaster, UK, September 07 – 11, 1997). Springer.

[16] Sandusky, R. J. and Gasser, L. Negotiation and the coordination of information and activity in distributed software problem management. In *Proceedings of the Proceedings of the 2005 international ACM SIGGROUP conference on supporting group work* (Sanibel Island, Florida, USA, 2005). ACM.

[17] Sonnenwald, D. H. and Pierce, L. G. Information behavior in dynamic group work contexts: interwoven situational awareness, dense social networks and contested collaboration in command and control. *Information Processing and Management*, 36, 2000 2000, 461-479.

[18] Star, S. L. The Structure of Ill-Structured Solutions: Boundary Objects and Heterogeneous Distributed Problem Solving. *Distributed Artificial Intelligence, Volume II*. Ed. L. Gasser and M. N. Huhns Morgan Kaufmann Publishers Inc., San Mateo, CA, 1989.

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

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

[21] Wenger, E. *Communities of practice: a brief introduction 2005* Accessed from http://www.ewenger.com/theory/index.htm on August 11, 2008.

[22] Ye, Y. and Kishida, K. Toward an understanding of the motivation Open Source Software developers. In *Proceedings of the 25th international Conference on Software Engineering* (Portland, Oregon, May 3-10, 2003). IEEE Computer Society.

[23] Zack, M. H. Interactivity and communication mode choice in ongoing management groups. *Information Systems Research*, 4, 3 1993, 207-239.