Int. J. Human-Computer Studies 77 (2015) 52–65

Contents lists available at ScienceDirect

Int. J. Human-Computer Studies

journal homepage: www.elsevier.com/locate/ijhcs

Revisiting Linus's law: Benefits and challenges of open source software peer review

Jing Wang*, Patrick C. Shih, John M. Carroll

College of Information Sciences and Technology, The Pennsylvania State University, University Park, PA 16802, United States

ARTICLE INFO

Article history:
Received 14 January 2014
Received in revised form
18 December 2014
Accepted 18 January 2015
Communicated by Francoise Detienne
Available online 28 January 2015

Keywords:
Online collaboration
Software peer review
Open source

ABSTRACT

Open source projects leverage a large number of people to review products and improve code quality. Differences among participants are inevitable and important to this collaborative review process—participants with different expertise, experience, resources, and values approach the problems differently, increasing the likelihood of finding more bugs and fixing the particularly difficult ones. To understand the impacts of member differences on the open source software peer review process, we examined bug reports of Mozilla Firefox. These analyses show that the various types of member differences increase workload as well as frustration and conflicts. However, they facilitate situated learning, problem characterization, design review, and boundary spanning. We discuss implications for work performance and community engagement, and suggest several ways to leverage member differences in the open source software peer review process.

1. Introduction

"Given enough eyeballs, all bugs are shallow" (Raymond, 2001). Linus's law highlights the power of open source software (OSS) peer review. As a high-profile model of large-scale online collaboration, OSS development often involves globally dispersed experts, mostly volunteers, collaborating over the Internet to produce software with source code freely available. Peer review is one of the core collaborative practices of OSS development: distributed participants evaluate and test the released software products, and report any problems they discovered or experienced; others jointly analyze and identify software defects or deficiencies, and generate solutions for repairing or improving the software products.

Large diverse communities are considered paramount to OSS peer review processes. "More users find more bugs because adding more users adds more different ways of stressing the program. […] Each one approaches the task of bug characterization with aslightly different perceptual set and analytical toolkit, a different angle on the problem" (Raymond, 2001). Extensive studies on OSS have shown the existence of other dimensions of member differences, such as heterogeneous motivations (Feller et al., 2005; Roberts et al., 2006), different expertise in software engineering and usability (Twidale and Nichols, 2005), and divergent perspectives (Sandusky and Gasser, 2005). The advances of social media provide opportunities for engaging an even larger audience in OSS development, and these potential contributors are likely to differ at even wider dimensions (Begel et al., 2010; Storey et al., 2010). Thus, understanding the role of member differences in the collaboration and social processes of OSS peer review, and particularly how it may be better leveraged is important for enhancing the understanding of OSS and online large-scale collaboration. However, little research has directly addressed diverse characteristics of members; existing work is largely focused on differences caused by roles (e.g., Daniel et al., 2013), distance (e.g., Cataldo et al., 2006), or national cultures (e.g., Shachaf, 2008).

To enhance the understanding of the OSS peer review process, we focus on the differences of participants and their impacts on the process, building on our previous study that has identified and characterized the common activities constituting the OSS peer review process (Wang and Carroll, 2011). We are especially interested in the less discernable or quantifiable attributes (e.g., informational and value diversity) in the OSS development context, rather than more readily observable ones (e.g., tenure within the site and the community, roles, language) as other studies did. To unfold the impacts of various types of differences, we conducted a case study of OSS peer review processes in Mozilla Firefox, a high-profile OSS project involving massive number of participants with a wide range of attributes. We analyzed member interactions recorded in bug reports, the central space for Mozilla's peer review. Participants who contributed to bug reports are valuable assets for OSS projects to retain, as they are probably more motivated than the generic Firefox users because using bug tracking systems to report, analyze, and fix bugs require more efforts than simply using the browser.

* Corresponding author. Tel.: +1 814 863 8856.
E-mail addresses: jzw143@ist.psu.edu (J. Wang), patshih@ist.psu.edu (P.C. Shih), jmcarroll@psu.edu (J.M. Carroll).

http://dx.doi.org/10.1016/j.ijhcs.2015.01.005
1071-5819/

Our findings indicate that informational diversity and value diversity result in both benefits and challenges to the work-related processes of OSS peer review as well as the well-being of open source communities. We disentangle the member differences that are often confounded with the prevalent dichotomic view of core/periphery and developer/user in current large-scale online collaboration literature. Such efforts create opportunities to understand and support overlooked groups of community participants, including triagers and co-developers. We also suggest implications for designing socio-technical interventions to mitigate the negative effects and augment the positive impacts of member differences. Distinct from prior research aiming at converting peripheral participation into active contributions, our design proposals offer an alternative way to embrace, leverage, and support member differences in communities that thrive on diversity.

2. Related work

2.1. Open source software peer review

OSS peer review is widely believed to be remarkably benefiting from a large community – "many eyeballs" – of members with different perspectives (Raymond, 2001). In general, the OSS peer review process begins with one submitting a bug report to the bug tracking system—an application that helps developers keep track of reported defects or deficiencies of source code, design, and documents. Others examine the defect causes and request additional information to determine whether the bug should be fixed. Once a solution is reached, they then commit a change set (mostly a patch) to the current software product. Our earlier work (Wang and Carroll, 2011) has codified the process as consisting of four common activities, including submission (i.e., bug reporting), identification, solution, and evaluation. These activities were externalized and made available in bug reports. They serve similar purposes as individual reviews, review meetings, rework, and follow-up in traditional software review, respectively, but fundamentally rely on web-based technologies. In addition to bug tracking systems in which people record and comment on bugs and issues, version control systems manage and synchronize committed software changes, while communication tools such as mailing lists and Internet Relay Chat (IRC) enable developers to discuss bugs.

Most studies related to the OSS peer review process were conducted from the software engineering perspective, deliberately modeling the information needs in bug report quality (Bettenburg et al., 2008a; Breu et al., 2010), inaccurate bug assignment (Jeong et al., 2009), efficiency and effectiveness of patch review (Rigby et al., 2008), and distribution of contributions in bug reporting and fixing (Mockus et al., 2002). Rigby et al. articulated stakeholders' involvement and their approaches for managing broadcast-based patch review (Rigby and Storey, 2011; Rigby et al., 2008). They also found that stakeholders interacted differently when discussing technical issues and when discussing the project scope.

With respect to the nature of a collaborative practice, much research effort related to OSS peer review has been devoted to explaining the coordination mechanisms (Yamauchi et al., 2000; Crowston and Scozzi, 2008; Sandusky and Gasser, 2005), negotiation (Sandusky and Gasser, 2005), leadership and governance (Fielding, 1999; Moon and Sproull, 2000), and the role of bug tracking systems (Bertram et al., 2010). Recent work by Ko and Chilana (2010) analyzed the reports of Mozilla contributors who reported problems but were never assigned problems to fix, indicating different competences of members in reporting bugs. Wang et al.'s analysis (Wang and Carroll, 2011) also showed large volume of bug reports failed to identify real bugs, increasing the cost of filtering them out. This study is to extend current understanding of OSS peer review by focusing on member differences of various attributes, particularly the impacts of these differences on the ways members interact and collaborate during the review process.

2.2. Diversity in collocated and distributed groups

Diversity is commonly defined as the differences of any attributes among individuals. As a complex construct, it has been studied in multiple disciplines, such as organizational behavior, sociology, and psychology. A complete review of this large body of literature is beyond the scope of this paper. However, regardless of variations between typologies, diversity can be of the readily visible attributes (e.g., gender, ethnicity, and age), of informational attributes (e.g., education, work experience, and knowledge), and of attitudes and values (e.g., whether members agree on what is important within the community and whether they have similar goals) (Van Knippenberg and Schippers, 2007; Jehn et al., 1999; Williams and O'Reilly, 1998). Diversity of an attribute can be further classified into three types—separation, variety, and disparity (Harrison and Klein, 2007). Separation refers to the differences in (lateral) position or opinion among members, primarily of value, belief, or attitude. Variety is the categorical differences, often unique or distinctive information, while disparity represents proportional differences along a continuum, mostly of socially valued assets or resources held among members. This conceptualization of diversity has important indication on the need of varying measurement when different types of diversity are being assessed.

Research on collaboration in collocated groups has a long history of analyzing diversity of various dimensions. Reviews and meta-analyses on this large volume of work suggested that the effects of diversity is contingent on the context: diversity can affect work processes, performance, and member identification in both positive and negative ways, and effects of the same diversity dimension may vary greatly across contexts (Harrison and Klein, 2007; Joshi and Roh, 2009). For instance, diverse perspectives tend to benefit work performance in short-term tasks, but these positive effects become much less significant in longer-term teams, and conflicts start to arise (Joshi and Roh, 2009). In general, a broad range of expertise and knowledge can enhance problem solving, decision-making, and even creativity and innovation, while differences of perspectives and values can result in dysfunctional group processes, conflicts, and poor performance (Milliken et al., 2003; Williams and O'Reilly, 1998; Van Knippenberg and Schippers, 2007). However, these studies were largely conducted in collocated groups in organizations or laboratory experiments.

A few other researchers looked into diversity in virtual teams. Shachaf (2008) explored the heterogeneity of national cultures of members from ad hoc global virtual teams at a corporation. The interviews showed such cultural diversity has positive influences on decision-making and negative influences on communication. Damian and Zowghi (2003) also focused on cultural diversity and found that it increased team members' difficulty in achieving common understanding of software requirements. Several chapters in Hinds and Kiesler (2002) discussed conflicts caused by differences of organizational cultures and informational diversity. However, such work still analyzed diversity in the settings in which group or organizational boundaries were clearly defined. There have been very few studies on volunteer-based large-scale online communities, such as OSS projects and Wikipedia, and therefore, they warrant additional examination.

Another theme of relevant research on virtual teams did not specifically analyze diversity but differences that were caused by distance, such as different information about remote contexts and different time zones. Unlike diversity literature focused on personal attributes, this body of work examined environmental factors, suggesting that dispersed locations led to conflicts (Cramton,

2001), communication and coordination (Herbsleb and Mockus, 2003; Cataldo et al., 2006; Olson and Olson, 2000), as well as uninformed decisions (Grinter et al., 1999). Our goal is to investigate member differences beyond the language and location dimensions, in order to understand how other aspects of variations may influence collaboration.

2.3. Member differences in large-scale online peer production

A crucial advantage of online peer production lies in its openness, engaging large-scale communities beyond a few experts. Members in such communities are likely to differ in various ways. Limited research has been done to articulate these variances and understand the challenges and benefits they bear on. The relevant studies fall into three major themes, *core* versus *periphery*, *developers* versus *users*, and *other types of variances*. We summarize them in the following subsections, respectively.

2.3.1. Differences between core and periphery

A body of research on large-scale online peer production touching on member differences was focused on contrasting core and periphery. One type of studies in this research theme modeled the structural characteristics in OSS development collaboration. These efforts distinguished core developers from peripheral members and indicated the relationship between project centrality and performance. Uzzi and his collaborators (Uzzi, 1997; Uzzi et al., 2007; Guimera et al., 2005) conceptualized social network structures of teams and their effects on collaboration in the organizational settings. Subsequently, researchers have applied social network analysis on collaboration among developers in OSS development. For example, Borgatti and Everett (2000) proposed an analytical model for detecting core and peripheral developers of OSS projects. Using a similar model, Tan et al. (2007) reported that both direct and indirect ties positively influence the productivity of teams, and the greater the cohesive ties that the team members form in their social network the more productive they are in OSS development. Adhering to this theme, Dahlander and Frederiksen (2012) investigated how one's position in the core/periphery structure affects innovation. Their findings suggested that an inverted U-shaped relationship between an individual's position in the core/peripheral structure and his/her innovation. In addition, spanning multiple external boundaries remains a positive delineator for most people, but is detrimental for the most core individuals. Crowston and Howison (2006) reported the emergence of a hierarchical structure through analyzing interactions around bug fixing of SourceForge projects. This hierarchy incorporated an additional tier, co-developers who submit patches, to the core/periphery dichotomy. They also found that the level of centralization is negatively correlated with project size, suggesting that larger projects become more modular. In general, OSS development is a highly decentralized activity, and research has found that there lacks a significant relationship between closeness and betweenness centralities of the project teams and their success in the majority of the OSS projects (Singh, 2010).

Aside from the effort of analyzing network structure, some other scholars characterized the distinct activities core and periphery performed in OSS development. In their widely-cited article, Mockus et al. (2002) reported the differences of contribution types and scale between core developers and the other contributors: a group larger than the core by about an order of magnitude contributed to defect fixing, and a group larger by another order of magnitude contributed to defect reporting. Rullani and Haefliger (2013) described the roles of core developers and those in the peripheries, and how the propagation of such standards is communicated through non-material artifacts such as code and virtual discussions as a social practice.

Our study disentangles the member differences confounded with status in the core/periphery structure, unfolding disparity of knowledge and separation of values (Harrison and Klein, 2007). One type of the disparity regards technical expertise, namely knowledge of programming and the code repository. Another type of disparity relates to process knowledge, which pertains to a specific community, such as the awareness of community norms, practices, and agenda. Core developers dominate the higher end of both spectrums, while the peripheral members spread over the other side. The separation of values is instantiated when core and peripheral members hold opposing beliefs of what is important to the software application. Conceiving member differences in these more nuanced ways beyond the dichotomic view of core and periphery create opportunities to further segment the participants involved in OSS peer review and to provide better support for their distinct needs. For instance, a group of people could fall at the higher end of process knowledge but lower end of technical expertise, like triagers. They have no intention to convert to core developers, but contribute significantly as coordinators and gatekeepers.

2.3.2. Differences between developers and users

Another group of researchers described OSS communities comprised of developers and users. Unlike the core/periphery structure that is commonly observed in other types of online communities, this developer/user distinction exclusively fits software development domain. Human–computer interaction scholars particularly emphasized the importance of users. Through examining usability discussions in OSS communities, they argued that end-users and usability experts could provide special expertise and knowledge the other developers do not have (Bach et al., 2009; Twidale and Nichols, 2005). In addition, Ko and Chilana (2010) studied user contributions to OSS bug reporting, suggesting that the valuable bug reports primarily came from a comparably small group of experienced and frequent reporters.

A few studies considered developers and users two different project roles. Barcellini et al. (2008) characterized how user-proposed designs were mediated between the user community and the developer community, identifying the emerging role of boundary spanners who participated in parallel in both communities and coordinated the discussions. A recent study analyzing 337 SourceForge OSS projects showed that variety diversity of project roles was positively associated with user participation (Daniel et al., 2013). Specifically, the number of users participating in a project is higher when members are more equally spread across developers and active users.

In line with thinking of member differences as variety, our work adds the diversity attribute of specific information about an issue to the attributes of technical expertise and project roles. For example, when describing a crash, members, regardless of being a developer or a user, probably can contribute information about different parameters of the system environment in which the crash occurs. Moreover, while the OSS peer review process is different from feature discussions and coding and development reported in Barcellini et al. (2008)'s article, we observed that a group of active volunteers, formally bug triagers, serve a similar role for bridging user and developer sub-communities to resolve bugs. We discuss the design recommendations to support this role, which contributes to the literature on role emerging design.

2.3.3. Differences of other attributes

A relevant group of studies provided indirect evidence of the existence of member differences; however, none was focused on elaborating these differences, particularly how the varying types of member differences impact the collaborative process. Early surveys on OSS revealed the variant motivations among individual participants. They subsumed utilitarian reasons (own needs for

software functions), joy and challenge, and reputation (Feller et al., 2005; Roberts et al., 2006). Work on negotiation and conflicts also hinted at different opinions participants might hold with respect to a wide range of issues. Sandusky and Gasser (2005) examined instances of negotiations in OSS bug finding and fixing processes in depth, showing how negotiation in different contexts affect the sense-making and problem-solving processes with 3 concrete examples. Research on Wikipedia, another high-profile example of large-scale online peer production, was mostly focused on characteristics of conflicts and negotiation among article authors. For instance, using history flow visualizations, Viégas et al. (2004) identified action patterns of users who had competing perspectives over article editing. Kittur et al. (2007) also used visualizations to model what properties of articles highly likely led to conflicts as well as cluster users into opinion groups.

The more direct efforts of analyzing member diversity centered on a few personal attributes that are relative easy to quantify, such as tenure, language, and activity characteristics. Besides project roles, Daniel et al. (2013)'s study assessed the effects of diversity of tenure, languages and activity levels on user participation in addition to variety of project roles. The results showed that tenure diversity (i.e., variance of the amount of time since one's registration at SourceForge) positively affect user participation, whereas variations in participants' activity levels was negative associated. The authors did not find any significant impact of language diversity on user participation. In the context of Wikipedia, Chen et al. found that increased tenure diversity (i.e., variance of the amount of time since one's first edit at Wikipedia) and interest diversity (i.e., variety of one's edits in different topics) increases group productivity and decreases member withdrawal, but after a point increased tenure diversity will increase member withdrawal (Chen et al., 2010).

Our investigation complements the prior studies with a focus on member differences of other types, particularly informational diversity and value diversity, which are relatively difficult to gauge in a quantitative way. Analysis on these dimensions also enhances the understanding of those more quantifiable diversity types, like tenure. For instance, member differences with respect to knowledge about the community relate to tenure diversity. We found these knowledge differences may increase workload of developers but create opportunities of situated learning for relatively inexperienced participants.

3. Methods

3.1. Case selection and description

To address the richness and complexity of the differences of massive participants in open source, we conducted a case study of Mozilla, a large and well-recognized OSS community. We focused our analysis on its core project, Firefox, expecting its end-user orientation would provide substantial instances of member differences, such as different expertise and experience.

The Mozilla community consists of both employees from Mozilla Foundation and Mozilla Corporation and volunteers. Core developers, mostly employees, are the ones who contribute significantly to the project. They are divided by a variety of different roles, including module owners, module peers, super reviewers, security group members, quality assurance members, and support team members. Each individual has to maintain and facilitate the evolvement of his or her specific areas. For example, Firefox had a sub-module Session Restore, which was supported by one sub-module owner and three peers. Some other developers also actively contribute to the project, but they are not affiliated to any of the organizations of Mozilla. Super reviewers assess significant architectural refactoring, any change to any API or pseudo-API, and all changes that affect how code modules interact. Peripheral members rarely participate in developing the software. They are comprised of end-users, web developers who design extensions or other applications on top of Mozilla's technology, network administrators, third-party developers, and developers from competitor products.

The peer review process in Mozilla follows a strict review-then-commit fashion. Any changes (e.g., patches) to a module or sub-module had to be reviewed by its owner or peers. Similar to other OSS projects, Mozilla's peer review involves a wide range of work and communication tools, including version control systems (i.e., Mercurial), bug tracking systems (i.e., Bugzilla), IRC, mailing lists and wikis. The bug tracking system is the primary space for peer review in Mozilla, unlike some other projects mainly relying on mailing lists for this purpose.

3.2. Data collection and sampling

Bug reports archived in Bugzilla – the bug tracking system Mozilla primarily uses for issue reporting, analyzing, and fixing – served as our major data source. Although our analysis focused on data from Bugzilla, our interpretation of those data was also informed by informal conversations with active contributors in the peer review process, examination of the design of Bugzilla, and relevant work artifacts that were shared publicly. These artifacts included 41 weekly meeting notes, 22 web documents describing work procedures and member roles, and 6 blog posts from individual members discussing Mozilla's peer review processes.

We retrieved bug reports created between two stable releases of Firefox from Bugzilla. This sampling strategy was intended to capture the possible behavioral changes near releases (Francalanci and Merlo, 2008). The retrieval was performed on July 28, 2010. It includes 7322 reports filed between the releases of Firefox 3.5 final (June 30, 2009) and 3.6 final (Jan 21, 2010). Additionally, we examined emails, blogs, wikis, documents and bug reports that were mentioned or given URLs in the bug reports we retrieved.

We used email addresses to identify contributors in Mozilla's peer review process. Core developers were defined as the ones listed on Mozilla's websites as module and sub-module owners and peers, Firefox development team members, super reviewers, security group members and members with additional security access, quality assurance team lead, and platform engineers during the time between Firefox 3.5 and Firefox 3.6 releases. 176 developers were identified in this category. Crowston and Scozzi (2002) suggested that it may be more useful to define core developers based on the amount of their actual contributions rather than their job titles, but our approach may better differentiate the values and knowledge of core developers from other volunteers.

Overall, the 7322 bug reports in our data collection were created by 5418 unique reporters. The number of bug reports created per reporter ranged from 1 to 82, with the median of 1 report per reporter. 1286 additional contributors participated by commenting in the discussion space of the bug reports. The number of bug reports contributed per contributor varied between 1 and 990, with the median of 1 report per contributor. Each bug report on average involved 2 Mozilla members in the discussion (median = 2; skewness = 6.765). It suggests that the peer review process was often a collaborative effort once a bug report was submitted, in which member differences are likely to surface.

To accommodate the possible variations of work processes and social interactions across bug reports, we performed stratified sampling with respect to bug status and resolutions. In Bugzilla, open bug reports had 4 types of status, *unconfirmed*, *new*, *assigned*, and *reopened*. *Unconfirmed* bug reports were bugs that had recently been added to the bug tracking system but not been validated to be true by anyone. In contrast, the other 3 types of open bug reports

Table 1
Distribution of the status of open bug reports.

Status	Number of bugs	% Of open bugs (%)	% Of all bugs (%)
Unconfirmed	2680	78.68	36.60
New	676	19.85	9.23
Assigned	39	1.15	0.53
Reopened	11	0.32	0.15
Total	3406	100	46.51

Table 2
Distribution of the resolutions of closed bug reports.

Resolution	Number of bugs	% Of closed bugs (%)	% Of all bugs (%)
Fixed	498	12.72	6.80
Duplicate	1346	34.37	18.38
Invalid	937	23.93	12.80
Worksforme	514	13.13	7.02
Incomplete	489	12.49	6.68
Wontfix	132	3.37	1.80
Total	3916	100	53.49

were all confirmed at least by someone, even though their validity might still be subject to others' judgment. Closed bug reports had 6 types of resolutions, *fixed*, *duplicate*, *invalid*, *incomplete*, *worksforme*, and *wontfix*. Tables 1 and 2 summarize the distribution of the status of open bug reports and the resolutions of closed bug reports, respectively.

For each of the 10 types of bug status and resolutions, we sampled 10% of them, which returned 732 bug reports in total. The sample size was just an initial estimation for starting our qualitative analysis, allowing us to perform in-depth examination instead of being overwhelmed by voluminous texts as well as to reach saturation. It eventually turned out to be a satisfactory estimation. For each type, we first intentionally selected the bug reports with number of comments at the 98th percentile in order to observe fairly complex social dynamics; then for the rest of the sample we randomly chose reports with fewer comments. All bug reports in Mozilla were created by a human reporter; they included bugs detected by automate tests, which were the cases for 70 bug reports in our retrieved data set and 21 sampled for the qualitative analysis. Thus, we treated all the bug reports the same when sampling them. Each bug report had its unique ID and often consisted of multiple comments. In the discussion space of each bug report, the first comment was labeled "description", which was generated by the bug reporter to describe the issue. All the other comments were indexed in order starting from 1. When quoting discourses in the next section, we use the format (bug ID; comment number; contributor type).

3.3. Data analysis

We carried out our qualitative analysis through three phases over the 732 bug reports with 8484 comments from 2742 contributors. First, the first and the second authors randomly selected 50 bug reports and read them separately. They discussed their observations and established shared understanding of the peer review process in Mozilla. Then during the second phase, the first author inductively coded the 732 bug reports, iteratively generated 628 codes occurring a total of 1623 times, and discussed and consolidated them with the other two authors during their weekly meetings. The frequency of occurrences we counted was on units that a code referred to. A unit was a complete episode demonstrating the impacts of member diversity on the OSS peer review process. It could be part of a comment. It could also span several comments, because participants sometimes had to comment on each other to clarify or elaborate their points in our study context. For example, for one episode we coded as "repeating how to avoid invalid bug reports" included two comments from the bug reporter and an active contributor, respectively. The reporter asked what information should be reported, and the response comment provided links to documented instructions. When one comment was divided into several units for different codes, we only assigned each unit a single code that best described the dynamics the unit demonstrated. Finally, 15 groups of codes that were pertinent to impacts of member differences and occurred the most frequently and deemed the most relevant to member differences were selected. Overall, the 15 groups of codes occurred 424 times accounting for 26.12% of total occurrences. These codes and themes are exclusive ones because of the way we analyzed the data. Therefore, none of the units were counted more than once for different codes. Then the authors integrated them into 6 themes. Table 3 summarizes the 6 themes with their definitions and 15 groups of sub-level codes with the number of their occurrences. We further describe each theme in Section 4, in which the quotes appear in the same order as the sub-level codes listed in Table 3.

We identified the existence of member differences and their various types through interpreting participants' discourses in the peer review process. The interpretation was inductive in the way that we did not bound it to a specific scheme but rather kept open to any unique information or different value statement emerged. Inferences generated during this process were subsequently consolidated with the guidance from the work of Jehn et al. (informational diversity and value diversity in particular) and Harrison and Klein (2007), Jehn et al. (1999). Adapting from their definitions of different types of diversity, we summarize the kinds of diversity emerged from our analysis in Table 4. For example, an episode in which a commenter stated he could not program and another commenter indicated to take over and submitted with a patch would be identified as disparity of proficiency in software engineering.

4. Results

Our qualitative analysis converged onto six major themes, unfolding the impacts of member differences on OSS peer review with respect to work performance and community engagement. These impacts indicate both challenges and benefits for OSS communities: challenges include increased workload and frustration and conflicts, while benefits range from situated learning, problem characterization, design review, to boundary spanning. They are associated with disparity and variety of expertise, information and resources as well as separation of values and beliefs among members. We describe these associations individually in the following subsections.

4.1. Challenges

4.1.1. Increased workload associated with informational diversity

Much workload of the peer review participants in Mozilla fell onto screening the large number of "non-real" bugs as Table 2 suggests. Our qualitative examination found that these bugs were largely associated with participants' informational diversity with respect to levels of technical expertise and experience with the standard software review practices and community norms. In our retrieved bug reports, nearly half ($n=3406$; 46.51%) lacked attention or actions (i.e., open bug reports) to move forward in the peer review process. On the contrary, the "non-real" bugs had already cost a significant amount of participants' efforts ($n=3418$; 46.68%). These bugs constituted 87.28% of the closed bug reports, including the redundant reports (*duplicate*), issues caused by reasons other

Table 3
Coding scheme and frequency of occurrences.

Theme (definition)	# Of occurrences	Sub-level code	# Of occurrences
Increased workload (overhead added to the core activities of the OSS peer review process)	80	Identifying duplicate bug reports due to different ways of describing bugs	59
		Repeating how to create quality bug reports	21
Frustration and conflicts (dissatisfaction with the outcomes and unresolved disagreement)	97	Feeling disappointed, underappreciated or claiming to switch browsers	67
		Stating values and beliefs to distinguish from the opponent	30
Situated learning (learning how to in practice)	89	Teaching how to test software and debug bugs	30
		Explaining community practices and norms	42
		Redirecting requests of patch review	17
Problem characterization (complementing and refining bug description and identification)	75	Complementing bug reports with testing results in different environments	48
		Refining problems with additional analysis	13
		Completing bug reports when original reporters did not respond	14
Design review (evaluating design ideas and implementations with different opinions)	27	Articulating additional scenarios and use cases	13
		Articulating design rationale with dissent	14
Boundary spanning (sharing information outside the discussion group)	56	Citing other sources outside Bugzilla	26
		Participating in discussion from 3rd parties	12
		Comparing with competitors' design	18

Table 4
Types of member differences emerged from analysis (adapted from Jehn et al. (1999), Harrison and Klein (2007).

Attribute	Types of differences	Definition
Informational	Disparity	(1) Differences in levels of technical expertise, i.e., proficiency in software engineering (2) Differences in levels of awareness of information and knowledge about the community
	Variety	1) Differences in domains of expertise regarding software design, e.g., usability, security 2) Differences in "batches" of information, resources or perspectives regarding a specific problem
Value	Separation	Differences in opinion or belief about what is important within the community and what the goal is

than Firefox (*invalid*), problems that cannot be reproduced by anyone other than the reporter or identified in the code (*worksforme*), vaguely described bugs (*incomplete*), and requests that will not be fixed (*wontfix*).

As the dominant of the "non-real" bugs, duplicate bug reports were generated when disparity of technical expertise among reporters existed. Sometimes the same problem can cause multiple symptoms, as Bettenburg et al. (2008b) also suggested in their analysis. Participants differ at their capabilities of identifying the cause and drawing connections between their own problems and existing bug reports. For example, a reporter could not figure out that his browser freeze was due to large numbers of checkboxes on the web pages as shown in the following quotes (Bug 506553). In contrast, another commenter pointed that out and was able to identify it as a duplicate of the bug "Session restore hangs/not responding with high CPU on large form with many checkboxes" (Bug 477564).

> Firefox 3.5 is locking on MusicBrainz artist pages that have a lot of releases. … The lock-ups also occur when using the parameter "-safe-mode". (Bug 506553; description; reporter/end-user).

> There is a login at the page and that might change the page itself. Does this page contain many checkboxes if you login? (Bug 506553; comment 1; commenter/active volunteer).

When reporting bugs, developers prefer to use very technical languages to describe bugs at the code level, in order to ease the communications with other developers. In contrast, end users with lower level of technical expertise tend to draft bug reports based on the observed symptoms. Such differences create barriers for non-technical people to find out whether their issues have already been reported and acted upon, leading to the generation of duplicate bug reports. For instance, the master report of a bug about displaying file icons was summarized as "nsIconChannel: GetHIconFromFile does not support Unicode filenames on Windows" (Bug 415761). In contrast, one of its duplicates described the issue in a language that is more familiar to the end-users.

> I've just noticed that in the Downloads menu (which shows when you download something) all icons are [shown] as folders. In previous version of Firefox if you download something in the Downloads menu it was shown (icons) – I mean Pictures had Picture icon, Files had File icon and so on. But now all items (pictures, files…) have the same icons – they are shown as folders. (Bug 504350; description; reporter/end-user).

Disparity of experience is another factor that the creation of the many "non-real" bugs, especially the invalid bug reports, can be attributed to. Participants who did not have much bug reporting experience tended to miss the testing step of using safe mode and a new profile to check if Firefox works before submitting their bug reports. Consequently, they filed reports that indeed described problems caused by extensions or third-party applications, which other experienced participants had to filter out.

> Try http://support.mozilla.com/en-US/kb/Safe+Mode and a new additional test profile http://support.mozilla.com/en-US/kb/Managing+Profiles. (Bug 507708; comment 1; commenter/active volunteer).

> [U]sing those modes I was able to determine it was the sk[yp]e plugin causing the issue (which was installed a few days ago). (Bug 507708; comment 2; reporter/end-user).

In summary, differences of technical expertise as well as of experience with the OSS peer review process and community practices increases the work of screening useless information from numerous bug reports. Such distribution of workload distracts developers from performing the core peer review activities, such as analyzing and fixing real bugs.

4.1.2. Frustration and conflicts associated with value diversity

The "many eyeballs" do not accomplish work as magically as Linus's Law sounds. We observed cases in which participants failed to reach any consensus after extensive negotiations. They experienced frustration and conflicts when they could not resolve the separation of their values and beliefs. Such differences occur when people vary in terms of what they think the group or community's goal and task should be, what is important within the group or community, whether members have similar goals, and so forth (Jehn et al., 1999).

Separation of values became salient in discussions on bug legitimacy and importance, which usually related to what was believed to be crucial to the community. Making such judgment cannot simply achieved by checking whether the software can function without fixing the bug, but rather entails personal assessments. Core developers emphasize how cost-effective a fix is code-wise, such as how many users a feature change could affect and how much the cost would be for code maintenance. In contrast, other users prioritize the usefulness of a feature to the individual user's task at hand and the consistency of interaction paradigm and interface layout. Whenever such differences arise and neither side can convince the other party, the core developers have the final say in the decision-making process and the end-users then get very frustrated or upset. In the following episode, users were arguing for keeping the "Properties" context menu because of its ease of use. However, core developers denied the request, explaining the gain of removing the corresponding code.

> When you have 50 images in the page, or even much less, you have ∗no∗ idea which URL points to the image you want. Till now, we had an easy way to get meta-data about images and links, and you remove this feature because you want to save 48 kb of code. (Bug 515368; comment 9; commenter/end-user).

> If you'll permit me to restate your point, I think you're saying that you estimate the maintenance cost of 1300 lines of unowned code and the UI cost of an extra context menu entry as being collectively lower than the value of providing more direct built-in access to image metadata on pages with many images than page info provides. Is that fair? …whether we think the feature serves a large portion of our users, what the cost is of keeping it, and what downstream impacts removal might have. (Bug 515368; comment 12; commenter/core developer).

> This is ridiculous and SO, SO unexpected to come from FIreFox. … (Bug 515368; comment 32; commenter/end-user).

Different beliefs of community objectives and cultures can even escalate to complaints and criticism on work processes, such as how decisions are made. Open source, originally emerged as an opponent of proprietary software, provided users an expectation of openness and an heightened ability to participate and contribute to the development process. Thus, users tend to conceive of OSS a democratic place to participate equally, whereas the ones who have been involved in OSS development for a fairly long time, such as core developers, consider themselves entitled more power in decision making. Consequently, users can easily feel unsatisfied when their requests are not fulfilled, and even decide to withdraw as illustrated in the last quote below.

> Sometimes the democratic system works. Sometimes a group with a very closed perspective hijack it. Congrats on making FF a laughing stock of a browser. (Bug 513147; comment 70; commenter/end-user).

> Mozilla is not a democracy. Open source does not mean "the developers must do my bidding". And being able to comment in Bugzilla is a privilege, not a right. (Bug 513147; comment 71; commenter/core developer).

> Mozilla is a meritocracy, and as such, the module owner/peers have final say over decisions concerning the codebase. (Bug 513147; comment 79; commenter/core developer).

> …there's not much I can do other than switch to Chrome until someone over there decides that our issue is worthy of their precious time. (Bug 533535; comment 22; commenter/end-user).

Additionally, frustration and conflicts also arose when people believed the participants had different goals and agendas. For example, developers perceived that some users contributed to bug reporting to get developers altering a Firefox module to fit their product websites or third-party browser extensions, which deviated from developers' goal of serving for the larger user population. Developers declined such requests but their responses were not well received.

> I'm not sure why you're coming here asking for help … without being willing to accept our analysis. (Bug 528153; comment 13; commenter/core developer).

In contrast, users believed developers' goal was just to minimize their own workload rather than helping with user problems. Accordingly, these users found it difficult to trust developers or appreciate their effort.

> I am also fully aware that in situations where people are doing technical support for a lot of people, their first objective is to keep their inbox clear by finding someone else to blame. (Bug 528153; comment 14; reporter/end-user).

In short, separation of values and beliefs entails negotiation of bug legitimacy, priority, community values, and norms. Failures to establish shared understanding and mutual appreciation lower the community morale, generate emotional conflicts, and even cause member withdrawal.

4.2. Benefits

4.2.1. Situated learning associated with informational diversity

Different levels of awareness of information and knowledge about the community do not just account for the generation of the many "non-real" bugs, but also creates opportunities for situated learning (Lave and Wenger, 1991). Such information and knowledge regards how to report and debug bugs, as well as community norms and practices. They are more experience-based compared to technical expertise, which often entails long-term formal education or training. Learning takes place simultaneously when participants collaborate on the peer review.

In addition to how to create quality bug reports described in Section 4.1.1, other knowledge being shared can be more difficult to detach from the specific context. We categorized episodes of communicating such types of information as situated learning to emphasize the context contingency, separating from the ones that can be easily standardized to alleviate developers' workload. The

following quote illustrates an experienced participant teaching the reporter how to create a test case specific to the bug.

> Can you simplify the test case, by just making the 'params' a hard coded string. I get your program now with bits and pieces. A proper bug report contains a standalone testcase.

> The POST method was invented before UTF-8. It is a proper unicode encoding, so it should not be a problem to send any Greek. (Bug 505250; comment 23; commenter/active volunteer).

Disparity of community awareness also emerges when experienced contributors correct other members' faulty operations or explain organizational norms and practices to clarify misunderstanding and confusion. People who are not regularly involved in Mozilla's peer review process probably know very little about, for example, what the meta-data of a bug report represents, but are only capable of reading with their own interpretations. In the following case, a reporter who had only commented on 1 bug in our data set believed the resolution of his bug should be flagged fixed because Firefox behaved normally after he upgraded to version 3.5.2. However, an active volunteer, who had contributed to discussions of 990 bugs in our data collection, modified this flag to *worksforme*, indicating the reason that FIXED was not an appropriate resolution flag.

> Resolution FIXED is reserved for bugs that had code applied to resolve the problem. So at least a bug number is needed. (Bug 502328; comment 19; commenter/active volunteer).

The situated learning not only regards sharing knowledge and norms, but also identifying appropriate experts. The awareness of who is qualified or available to address the request requires extensive and long-term involvement in the project. Participants without such awareness often specify developers that are not suitable to resolve the bug. In these cases, well-informed members will redirect the request, demonstrating who the right person is through action.

> I can't review code in js src. I'm forwarding to [developer 1], feel free to forward again in case i did a bad choice. (Bug 526422; comment 2; commenter/core developer).

In sum, information and knowledge about the community is highly related to the context. Members' different levels of awareness of them create opportunities for inexperienced members to learn from experienced collaborators through practice. Enjoyable learning experiences may enhance the commitment of community newcomers.

4.2.2. Problem characterization associated with informational diversity

One of the advantages of OSS peer review lies in the variety of information and resources the "many eyeballs" have. Such diversity does not just increase the chance of finding more bugs as Raymond elaborates on Linus's Law (Raymond, 2001); we observed that it also enhanced the characterization of a bug that had already been reported. One recurring pattern of problem characterization was to corroborate and enrich the report with testing results in other environments. Users contributed in a fairly simple and standard way rather than had to acquire certain analytical skills: they shared the information regarding the operating systems, software versions, and other configurations in which they experienced the bug. An individual usually had much more limited access to those various testing resources than the crowd. Moreover, users varied in the ways they used the software, habitually or accidentally, encountering the problem through slightly different paths. Such differences enabled a bug report to be collectively constructed with improved accuracy, thereby facilitating diagnosis and the peer review process. The role of bug reporters extended beyond the ones submitting reports to bug tracking systems, encompassing whoever complemented the information about a problem or an issue. The following excerpt illustrates different participants contributing information regarding the various operating systems and localizations (i.e., software adaptation to different languages, regional differences, and technical requirements of a targeted international market).

> Firefox 3.5 does not ask for save all tabs even when configured to do so and even when there is only one Firefox window opened, … My Firefox is 3.5 Brazilian Portuguese version. My Windows is Vista 64 Ultimate. (Bug 502532; description; reporter/end-user).

> I have a similar problem. Mine does ask, and I tell it to save my tabs. I open it up the next time, and I get my home page, not my saved tabs. I tell Firefox to clear the following data when I close it: Browsing History, Download History, … I'm running Firefox 3.5 English and Windows Vista x64 Ultimate. (Bug 502532; comment 1; commenter/end-user).

Aside from variety of information, different levels of technical expertise (e.g., debugging) also appeared to help problem characterization when participants with even slightly better expertise generated hypothesis, pruned irrelevant information, and performed additional investigation. These extra analyses refined the problem characterization, allowing someone else to eventually identify the cause of the defect. In the following episode, the commenter described how he narrowed down the reported problem through a series of simple hypothesis testing, which the bug reporter did not perform but only described the symptom of Firefox failing to load any web page.

> After clicking on any link to a website, and waiting for about 15–30 s, I will get a "Server not found" that Firefox can't find the server. I would have to refresh the page in order to get the website loaded. (Bug 502963; comment 0; reporter/end-user).

> [I] face the same issue. … i suspected it's a network connection, but other tabs would be loading without issue. chrome doesn't face the issue. i don't face this on ubuntu. i started facing this only in 3.5. it's most readily seen while stumbling. and it's not a traffic issue (i rarely have more than 3 or 4 tabs open). (Bug 502963; comment 2; commenter/end-user).

Compared to the relatively simple debugging techniques (e.g., altering some environmental parameters), some participants even demonstrated higher level of technical expertise in minimizing test cases, conducting systematic analysis, and capturing technical details. In a bug about Firefox halting on large amount of data (Bug 506844), for example, a commenter created a test case to enable other developers to reproduce the bug. He shared the quantitative results of his preliminary analysis, concluding that "the slowdown is exponential to the size of the old content (the contents that is erased), but doesn't depend (much) of the size of the new contents."

The other pattern of problem characterization occurred when participants exhibited variety of time resources and commitment. In cases where a contributor was not able or willing to take the time and efforts addressing certain requests or inquiries, other contributors might do so to complete the problem characterization. In the episode below, a reporter did not provide a test case in his post more than one month ago. An active contributor had asked for a test case to analyze the problem but the reporter never responded. After a few weeks, another participant created and uploaded a workable test case.

> 2009-07-18A test case is far more useful, then a crash report … Can you attach input to this bug, that crashes FF? Of course, a

full test case would be better … (Bug 506844; comment 2; commenter/active volunteer).

2009-08-28 Since https://bugzilla.mozilla.org has banned our country IPs I could not check my reported bug regularly. (Bug 506844; comment 12; reporter/end-user).

2009-09-16 Created attachment 401081.

This demonstrates the slowdown on emptying a div with a bunch of contents via innerHTML … Using the testcase: (1) fill the div with either 250, 500 or 1000 lines. (2) Empty it with "Empty (innerHTML)" or by filling it with any number of lines. (Bug 506844; comment 14; commenter/end-user).

Overall, despite the noise that "many eyeballs" introduce to the bug finding process, OSS peer review still benefits from the diverse community for effective characterizations of software defects and deficiencies. The OSS peer review process allows individuals to complement and refine the bug reports with variety and disparity of information and resources, facilitating the process of problem characterization.

4.2.3. Design review associated with informational diversity

The large number of end-users in the Mozilla community not only generates the disparity of technical expertise and resources pertinent to the software development, but also increases the variety of domain expertise and perspectives. The rest of the community also varies in terms of the areas they are specialized in or issues they know well. Such differences allow early design ideas and actual design implementations of bug solutions to be reviewed by different experts, creating opportunities of design improvement.

Users are the best for providing first-hand knowledge of how the software will be used, which adds to the domain expertise in usability and user experience. Even though they may not always be able to reflect or explain their experiences or behaviors, they can still articulate the scenarios or use cases that are overlooked or misconceived by the developers. In the bug report that proposed to "improve Mac install experience: detect that we're running from disk image or Download folder", a participant evaluated the design ideas from the designer and found a special use case of the Firefox installer was missing.

Another thing to consider, if only for being aware of it, is that users with encrypted home folders (FileVault) will be copying and not moving Firefox when they drag it to the Applications folder (since the home folder is an encrypted volume different from where the Apps folder reside, and drag&dropping to another volume defaults to copying instead of moving). (Bug 516362; comment 12; commenter/end-user).

Given the complexity of OSS like Firefox, the implications of a design implementation could be very complicated and require knowledge and expertise from different fields, which is beyond an individual's ability to comprehend. Elaboration and dissent among parties with diverse perspectives help articulate rationale of software design, improving its quality. Such disagreements, or even critiques, are often conveyed in a concrete and constructive way. For instance, Firefox has teams specialized in user experience, platform, security, localization (L10n), and so forth. Members from different teams tend to evaluate design from their own lenses. In the bug for speeding up Firefox startup, developers believed the file *browserconfig.properties* "*wasn't doing anything important*", and implemented a patch to move this file into a jar. They contended that this solution could avoid "*too much compat cost, and pain if we need to switch back later*." However, a developer from the localization team, who worked on "translating an application's menus, dialogue boxes, labels and other settings (font, character encodings, dictionaries, default search engine etc) for a particular locale", disagreed with the design approach after evaluating the patch and articulated its downsides from the localization perspective.

FWIW, I don't agree with the initial comment in this bug, setting the homepage ∗is∗ important. It's a revenue-generating asset for us and for everybody that can spoof it, so making that hard to do is good… browserconfig.properties contains locale dependent information, and should not be in browser.jar. … We're also contemplating changing the url [of browserconfig.properties] completely for some of our smaller locale, in particular for Kurdish google doesn't seem to be a good choice as it prompts users of our latin-based kurdish with arabic script kurdish. (Bug 524201; comment 13&15; commenter/core developer).

Patch review policies in Mozilla only authorize module/submodule authors to evaluate changes to their own code in order to ensure the proper expertise of the reviewers. However, this also means that these designated patch reviews typically only involves one developer per patch, which is sometimes not enough to capture all the possible defects. It is not uncommon for contributors outside of the designated review groups to capture faulty designs. The episode below shows that the requested reviewer missed one defect in a patch, which was caught by an outside developer.

The patch is also wrong because bookmarksBarContent may be absent from the document. (Bug 504858; comment 30; commenter/core developer).

Indeed! I should have caught that… (Bug 504858; comment 31; commenter/core developer).

To sum up, variety of domain expertise and perspectives, both in general and specific to a bug, enriches the scenarios and use cases being articulated at the early stage of design. Furthermore, it facilitates the articulation of design rationale through dissent and critique, assuring the quality of implementations.

4.2.4. Boundary spanning associated with informational diversity

Informational diversity of participants in OSS communities also includes the variety of knowledge and resources spanning group and organizational boundaries. Similar to boundary objects (Star, 1989), participants share or leverage knowledge and resources from other work groups, sub-communities, and organizations that would not otherwise be accessible or obtainable by the developers involved in resolving a bug.

In Mozilla, contributors cited reports and discussions outside bug tracking systems beyond the current discussion group, such as user support forums and informal community websites, to confirm a bug's validity.

I'm gonna confirm this then because I seen a report on mozillazine with a trunk build that this happens. (Bug 501904; comment 4; commenter/active volunteer).

Although this was often employed as a strategy to draw developers' attention and entice them to act on the problem (e.g., Bug 501413), it sometimes provided supplemental information that facilitated problem analysis (e.g., Bug 527540).

Perhaps this seems like a small problem in uncommonly-used UI, but apparently regular users use the bookmarks sidebar more than we realize: there have been numerous reports of this problem in the community forum in Israel and IRC #mozilla.il since the release of Firefox 3.5. (Bug 501413; comment 20; commenter/core developer).

… adding additional versions to the bug summary based on the interesting-modules-with-versions report in http://people.mozilla.com/crash_analysis/; I'd note that the versions reports show that the older versions seem to be ok, but the newer versions seem to be causing crashes. (Bug 527540; comment 10; commenter/core developer).

Knowledge regarding third-party applications is also critical to bug analysis because of the high interdependencies among software applications. For instance, Firefox is dependent on a variety of existing third-party software infrastructure, including database, operating systems, and multimedia platforms. At the same time, its design also affects extensions and websites. Developers of these third-party applications are actively involved in Firefox's peer review process, communicating and collaborating with other participants on diagnosing and resolving issues affecting both sides. The excerpt below illustrates a situation in which the reporter encountered the bug, "Browser segfaults attempting to delete a database entry (in do_lookup_x () from/lib/ld-linux.so.2) on Twitter and others." A developer from Red Hat (software company primarily offers operating systems and middleware), which was the relevant third-party in this problem, shared the link to a similar report at Red Hat that offered his analysis and possible solutions.

We received a similar crash report at https://bugzilla.redhat.com/show_bug.cgi?id=560662 (Bug 501992; comment 4; commenter/third-party contact).

[Reporter]: your problem is caused by canberra which is presumably delivered by your vendor (fedora), please use [comment 4]'s bug link to deal with this problem. (Bug 527029; comment 6; commenter/core developer).

Other than cooperating with third-party contacts, participants in Firefox peer review collaboration also observe, compare, and learn from other browsers, such as Microsoft Internet Explorer, Google Chrome, and Opera. Participants with experiences using multiple browsers often report their personal experiences with other browsers back to the peer review community.

I believe that the "Paste and Go" option in both Opera and Chrome is something that would greatly benefit Firefox. For those that are not familiar with what I am talking about, on both of the aforementioned browsers, if you right click on the address bar one of the options that appears in the menu alongside "Cut", "Copy", and "Paste" is the "Paste and Go" option. This is a small time saver that belongs in Firefox. (Bug 501558; description; reporter/end-user).

More interestingly, developers of competitor browsers occasionally join the conversation and share their design knowledge. The following episode describes a case in which a Chrome developer shared the implementation details of the Chrome browser when he observed that Firefox developers were trying to match Chrome's perceived scrolling speed. This also indicates that developers of major browsers are monitoring features of competing products closely, and sometimes they even collaborate to make their products better.

Just so you guys know, if the goal was "match Chrome behavior", doubling the system scroll lines value isn't Chrome's behavior at all. (I wrote Chrome's behavior.)… In short, we use 100 pixel per "three lines", and the system ticks→lines mapping (which defaults to three lines per tick, so by default we do 100 pixel per tick). (Bug 513817; comment 89; commenter/competitor developer).

In short, the fluidity of OSS organizational boundaries and the flexibility of OSS community structure allow information flow and resource configuration spanning across groups and organizations. Participants with multiple identities contribute their diverse knowledge and resources that would otherwise be inaccessible to the peer review process. Such differences benefit problem diagnosis and design idea generation.

5. Discussion

Our investigation suggests that informational diversity and value diversity generate both benefits and challenges to the work processes and social processes involved in the OSS peer review. For work process activities such as bug submission, problem identification, solution generation, and solution evaluation that are common to the OSS peer review process (Wang and Carroll, 2011), variety and disparity of expertise, information and resources enable the crowd to complement bug reports with their own experiences and analysis. The crowd enriches the articulation of use scenarios and design rationale through critique and dissent, as well as exchanges information with external groups and organizations. Other than enhancing problem characterization, design review, and boundary spanning, member differences also create challenges to the work process of OSS peer review. Specifically, disparity of technical expertise and community awareness leads to a large number of redundant, invalid and ambiguous bug reports submitted, increasing the workload of screening bug reports and repeating to novices about standard peer review techniques. Such disparity, on the other hand, is beneficial to the social processes in the peer review. It creates opportunities for cultivating novices with contextualized software peer review approaches and organizational practices, reinforcing shared norms, and building social connections. These practices contribute to community sustainability when facing member turnover. Besides the benefits of situated learning, separation of values and beliefs stimulate negotiations over bug legitimacy, priority, and community objectives. Such differences create challenges of frustration and conflicts, which may discourage community engagement or even result in member withdrawal.

5.1. Understanding member differences in OSS development

Our study contributes to research on large-scale online collaboration in several ways. First, we untangle the underlying differences among members that are confounded with the focus on core/periphery dichotomic prevalent in large-scale online peer production literature. Isolating disparity of technical expertise from that of community awareness, for both of which core developers are at higher levels, provides opportunities to uncover the overlooked groups of contributors. Participants with high-level awareness but low-level technical expertise, like triagers, and those with low-level awareness but high-level technical expertise, like co-developers are such groups. Better understanding of their distinct needs and activities may lead to tailored design that supports their identities and experiences. Second, we refine the characterization of informational differences between developers and users, dividing them into disparity and variety. Ko and Chilana (2010) emphasized the cost caused by disparity of knowledge and experience, suggesting to recruit more talented volunteers. Our observation similarly indicates that such disparity increased workload; however, the variety of expertise, information and perspectives enhanced problem characterization, design review and boundary spanning. Conceiving of variety of information at the level of specific information regarding a specific issue calls for thinking and designing beyond the project roles of developer/user, extending to activity roles. Additionally, the disparity of technical expertise and community awareness also facilitated situated learning in the community. Third, we have studied additional diversity attributes that were not examined in previous online peer production studies. Prior work was

largely focused on quantifiable attributes, such as tenure, language, and interests. Our qualitative analysis captures the relatively implicit and dynamic ones, for instance, variety of information and resources regarding a specific bug.

Our work also extends the understanding of diversity with findings from the context without traditional organizational boundaries. The impacts of informational diversity we identified partially accord with previous literature on diversity, which suggests enhancement on work performance (Harrison and Klein, 2007; Van Knippenberg and Schippers, 2007). We have specifically discovered that variety of information can serve as a resource for problem characterization, design review, and boundary spanning. But these benefits manifest themselves in the OSS community differently from workgroups with clearly defined group or organizational boundaries: configuration of diverse expertise, information and resources is not pre-defined but highly fluid. Furthermore, in contrast to prior research, our results also indicate the challenges associated with informational diversity, particularly disparity of technical expertise and community awareness, which may increase members' workload of filtering bug reports and communicating standard procedures. Such disparity is also somewhat relevant to tenure diversity discussed in earlier studies, as old-timers are likely to have higher level of community/organizational awareness (Daniel et al., 2013; Chen et al., 2010; Williams and O'Reilly, 1998). In contrast to the positive effects reported in those papers, we observed more complicated and nuanced impacts. Different levels of awareness of information and knowledge about the community can increase the workload of filtering bug reports and communicating standard procedures. Meanwhile, unlike technical expertise entailing relatively long time to acquire, some contextualized peer review techniques and community practices are easy to communicate through online conversations, which generate learning opportunities and facilitate community development. With respect to value diversity, our findings are in line with existing understandings. In OSS peer review, conflicts and frustrations arise when members have different beliefs of what is important to the community and what their goal is. Such differences are not tied to organizational boundaries like demonstrated in the literature (Hinds and Kiesler, 2002; Shachaf, 2008) but largely associated with the sub-communities participants identify with (i.e., developer vs. user sub-communities) or their status (i.e., core vs. periphery).

We also want to distinguish our contributions from efforts to evolve peripheral members to the core in online communities: the benefits and challenges of member differences characterized in our study indicate that converting members from peripheral to core may not be the only or ultimate goal for designing effective online communities. Instead of reducing heterogeneity, we propose designs embracing and supporting the ways members differ, while providing possibilities for member integration and evolvement.

5.2. Design implications

5.2.1. Enabling alternative representations of bug reports

The challenge of increased workload and the benefit of problem characterization entail special support for peripheral members, who do not contribute regularly but constitute the majority of the community. Moreover, people with different domain expertise could become a resource for knowledge development in the communities. Current bug tracking systems are largely designed to support core developers' work. Providing alternative representations of reported issues can increase the visibility of parallel ontologies, which may mitigate member differences.

One approach to reducing submissions of duplicate reports is to translate archived reports and ongoing resolving progress in an end-user-accessible way. For example, for reports or code changes regarding user interface, bug tracking systems can represent them in an interactive visualization that groups and displays them with each visible component of the browser interface, similar to the implementation of LemonAid that enables users to find help through directly selecting interface elements in question (Chilana et al., 2012). Whenever someone wants to create a report, s/he can hover over the browser component s/he has problems with to obtain a list of references to the issues filed under this component. To translate such visual input into technical description, an additional group of contributors like active volunteers or bug triagers can help bridge the gap. This will help narrow down the duplicate candidates, though this alternative representation of bugs may not apply to issues that are embedded in components that are invisible on the browser interface.

Another approach to improve identification of duplicate reports before submission is to enable flexible creation of meta-data that bridge the gap of contributors' varying use of terminologies to characterize a problem. Other than providing system-defined classifiers to maintain consistent structure within the code repository, allowing user-generated vocabularies can accommodate a greater variety of mental models of the software application. Tags may be a good design option for this purpose. Although they are usually personal and not always informative, tags can still serve as an extra index for searching in the system. Moreover, participants who were to submit a bug report but found out the issue had already been reported may add their interpretations of the problem as tags. Such accumulation can further increase the likelihood of identifying duplicates through bug search. Additional analysis and visualization of the tags, such as aggregation and highlights, may also inform the core developers of how end-users perceive problems and issues that they deem to be more important.

In addition to preventing from filing duplicate bug reports, the ability to consolidate information from different reporters is equally important. As our results show, different reporters do not always report identical information on the same issues. Offering cross-references to duplicate reports is a simple way for aggregation, but lacks some integration. Tags may be an alternative: they are easier to be aggregated than bug reports; they would be useful if each of them is informative and distinctive.

Reducing the creation of invalid bugs can also address the increased workload of filtering reports. Users usually do not read through documentations that provide extensive instructions about how to participate in OSS peer review process, either because they are not willing to or because they are not aware of these documents. That is probably why more experienced contributors must repeatedly ask the novice contributors to repeat test cases in the safe mode and under a new browser profile. Minimizing the instruction to guide reporters to differentiate user support issues from real bugs before they submit a report may be an effective design. For instance, rather than archiving instructions in other community sites, a page before the report submission form can be designed in the bug tracking system, asking the reporter to try to reproduce the problem in safe mode or with a new profile.

5.2.2. Enhancing community awareness and learning experience

Differences of experience may delay the progress of bug analysis and fixing because of the overhead for communicating peer review techniques and community practices, as well as redirecting inappropriate patch review requests, even though they open up chances of situated learning. Reducing the overhead requires long-term involvement or at very least – long-term observation – in the peer review process. However, this requirement contradicts the motivation of many contributors who just want to get their own problem solved. Therefore, making that experiential knowledge easy to access can enhance community awareness and learning experience.

Although tutorials and documentation of common techniques and community standards are available to public, they are not activated in inexperienced contributors' actual participation. Automate reminders in situ may help reduce the workload of core members. This may increase the efficiency of peer review, but may also decrease interactions between newcomers and old-timers and inhibit knowledge sharing within the community. Thus, an alternative solution is to nurture a larger group of active members who are not core Mozilla developers to mediate expert matching. These contributors are motivated to maintain active and long-term involvement in the entire OSS peer review process. They do not have to acquire as much domain expertise as the core developers, but they have more advanced knowledge of community practices and refining bug classification than the mass peripheral. These active volunteers can also introduce low-risk and easy tasks to novices because of their familiarity with the difficulty and severity of the ongoing project work. To sustain this group of contributors, the community needs to show appreciation and acknowledge the contributions of these triagers, which has not been the case based on our observations. One possible improvement is to make the responsibilities and the role of these triagers explicit, such as organizing a formal team to work specifically on orienting newcomers and prioritizing bugs, turning the invisible work to visible (Suchman, 1995). It may also be beneficial to expand the types of work this group of people do to distinguish them from the rest of the community. For instance, other than teaching peripheral members and referring relevant bug reports to core developers, they can also help build social connections between end-users and core developers.

Aside from cultivating the learning community from peripheral members, facilitating the evolvement of this group of active volunteers is also important. They are probably motivated to develop their technical expertise, which is not just beneficial for their individual career but also for the community if an expert left. For instance, promoting a mentoring program can help these contributors to identify their interests and match them with experienced developers who are able to provide them with specific advice. Alternatively, designing technologies to aggregate or externalize core developers' activities may also create easy access to observe how experts work and increase the likelihood of social interactions. Such learning experience may enhance member identification with the community, and thus retain long-term membership and sustain long-term contribution.

Externalizing the knowledge of developers' specialties and status, which was implicit and only accessible to co-developers in the Mozilla community, may assist with finding the appropriate experts. One possible approach is to construct dynamic profiles that automatically reflect developers' activities on software products as well as other people who have interacted with them over these products. Another option could be visualizing the social networks of active community members based on their activities. Adding a social feature that shows the volume and status of developers' work in the community can also help other participants decide whom they should ask. Developers of Mozilla appropriated their user names in Bugzilla to indicate the time period they will be unavailable, such as being away on vacation. This confirms the need of such a feature.

5.2.3. Establishing creditability and incentives

The challenge of frustration and conflicts implies that participants do not believe in the legitimacy of each other's expertise. Bach et al. reported similar findings in their analysis on expertise in usability (Bach et al., 2009). The belief is also critical to effectively integrating critiques raised in design review. Mozilla acknowledges volunteers who have made significant contributions by announcing their names in its weekly project meetings and core developers' blogs. It also credits the ones who have made enough quality contributions with privilege of editing bug reports in Bugzilla. However, this coarse characterization of contributions is weak and insufficient.

Enhancing the visibility of contributions may facilitate creditability establishment. One design solution is to create profiles for individuals that record past contributions and displaying them when the participants interact with others. Tracking the accumulation of experience and quality work may also enhance participants' self-efficacy in producing valuable contributions to the community, motivating them to continue participating.

Additionally, a sophisticated evaluation system can address the variety of contribution types and motivations of participation. Currently, the community only maintains a web page that lists contributors who are identified by whether they wrote code or documentation or participated in testing tasks for Mozilla. A finer characterization of contributions would be more beneficial for rewarding participants. For example, a bulletin board that specifies different types of contributions and updates representative work for each type might be a design option to present quality of contributions rather than quantities. Furthermore, the bug tracking system may also allow contributors to rate their own contributions. A contributor can rate a comment highly if s/he finds the information useful for bug analysis. The contributors who have provided many highly rated contributions could receive special recognition in the community as a reward.

5.2.4. Creating channels for information integration

Third-party contacts that bridge different work groups and connect OSS communities with other organizations do not participate regularly in OSS communities. Probing them only when issues arise is not always an effective practice. In contrast, keeping regular contact with them may augment the benefit of boundary spanning, forming early design ideas and foreseeing potential problems.

Leveraging new media, or even creating a new communication channel for debriefing the most recent updates of other groups and organizations may facilitate information exchange. Such information are currently scattered in various places and information sources, and most members do not have a clue of how to gather that. The integration of information sources provides the basis for content synthesis. Furthermore, updates like this do not have to be frequent or overly detailed but classifiable by multiple criteria beyond organizational boundaries. Have integrated information sources could also reduce redundant dyadic interactions. Mozilla currently hosts weekly project meetings, in which the user support team raises popular concerns or complaints in the support forum to the development team. It also implemented a blog feed that incorporates relevant blogs within the Mozilla community where members share their work progress and reflections in their blogs.

5.3. Limitations and future work

Our current analysis is constrained by its case selection and data sources. We chose Mozilla because of its large and diverse community, complexity and recognition; however, some of its characteristics differ from other OSS projects, which could be affected differently by the types of member differences described in this paper. Unlike purely volunteer-based communities, Mozilla is comprised of both paid members and volunteers. Moreover, Mozilla's OSS practices may differ from those of smaller and decentralized OSS projects; it follows strict and formal procedures of peer review and clearly defines the roles and responsibilities of its core members. We focused our investigation on bug reports archived by bug tracking systems, even

though the interactions in the peer review process may involve the usage of other computer-mediated communication tools. Other OSS projects that heavily rely on mailing lists to perform peer review may exhibit different impacts of member differences. Rather than calibrating the degrees of member diversity, our qualitative analysis was intended to identify and characterize the impacts of member diversity, particularly the dynamics emerged from member interactions. This approach determines that the types of member diversity were inferred from observations of those dynamics. We plan to compare our findings with other OSS communities in future work and gather feedback on our design implications from the Mozilla community to evaluate their feasibility and effectiveness.

6. Conclusion

Member differences are inevitable and important to effective collaboration and innovation. Large-scale volunteer-based distributed online collaboration requires understanding of differences among participants and their impacts. Our investigation on the peer review processes in an OSS community, Mozilla, shows that member differences bring both challenges and benefits to the online collaboration. Disparity of information, particularly with respect to technical expertise and community awareness, increases members' workload but facilitates situated learning. Variety of information and perspectives regarding a specific issue as well as variety of domain expertise is beneficial to problem characterization, design review, and boundary spanning. Moreover, the challenge of frustration and conflicts arises when separation of values is externalized through communication. These findings complement the understanding of diversity studied on ad hoc teams in organizational and laboratory settings. Our articulation of member differences also explicates underlying variances among community participants with greater granularity than the prevailing dichotomy of core/periphery or developer/user in online peer production research. It creates opportunities to better understand, support, and leverage insufficiently recognized groups of community members, such as triagers and co-developers. Our study also suggests that leveraging crowds to collaborative knowledge work not only lies in evolving the peripheral to the core, but also appreciating and supporting the heterogeneity. We recommend approaches to augment the benefits as well as mitigate the challenges—enabling alternative representations of bug reports, enhancing community awareness and learning experience, establishing creditability and incentives, and creating channels for information integration.

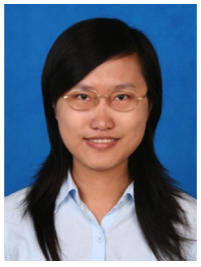
Acknowledgements

This work is supported by the US NSF (0943023). We thank our partner, the Mozilla organization for sharing their practices.

References

Bach, P.M., DeLine, R., Carroll, J.M., 2009. Designers wanted: participation and the user experience in open source software development. In: Proceedings of the CHI2009. ACM Press, 985–994.
Barcellini, F., Detienne, F., Burkhardt, J.M., 2008. User and developer mediation in an Open Source Software community: boundary spanning through cross participation in online discussions. Int. J. Hum. Comput. Stud. 66, 558–570.
Begel, A., DeLine, R., Zimmermann, T., 2010. Social media for software engineering. In: Proceedings of the FSE/SDP 2010. ACM, 33–38.
Bertram, D., Voida, A., Greenberg, S., Walker, R., 2010. Communication, collaboration, and bugs: the social nature of issue tracking in software engineering. In: Proceedings of the CSCW2010. ACM Press, 291–300.
Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R., Zimmermann, T., 2008a. What makes a good bug report? In: Proceedings of the FSE 2008. ACM Press, 308–318.
Bettenburg, N., Premraj, R., Zimmermann, T., Kim, S., 2008b. Duplicate bug reports considered harmful…really?In: Proceedings of the ICSM08. IEEE, 337–345.
Borgatti, S.P., Everett, M.G., 2000. Models of core/periphery structures. Soc. Netw. 21, 375–395.
Breu, S., Premraj, R., Sillito, J., Zimmermann, T., 2010. Information needs in bug reports: improving cooperation between developers and users. In: Proceedings of the CSCW2010. ACM Press, 301–310.
Cataldo, M., Wagstrom, P.A., Herbsleb, J.D., Carley, K.M., 2006. Identification of coordination requirements: implications for the design of collaboration and awareness tools. In: Proceedings of the CSCW2006. ACM, 353–362.
Chen, J., Ren, Y., Riedl, J., 2010. The effects of diversity on group productivity and member withdrawal in online volunteer groups. In: Proceedings of the CHI2010. ACM Press, 821–830.
Chilana, P.K., Ko, A.J., Wobbrock, J.O., 2012. LemonAid: selection-based crowdsourced contextual help for web applications. In: Proceedings of the CHI2012. ACM Press, 1549–1558.
Cramton, C.D., 2001. The mutual knowledge problem and its consequences for dispersed collaboration. Org. Sci., 346–371.
Crowston, K., Howison, J., 2006. Hierarchy and centralization in free and open source software team communications. Knowl. Technol. Policy 18, 65–85.
Crowston, K., Scozzi, B., 2002. Open source software projects as virtual organisations: competency rallying for software development. IEEE Software 149, 3–17.
Crowston, K., Scozzi, B., 2008. Bug fixing practices within free/libre open source software development teams. J. Database Manage. 19, 1–30.
Dahlander, L., Frederiksen, L., 2012. The core and cosmopolitans: a relational view of innovation in user communities. Org. Sci. 23, 988–1007.
Damian, D.E., Zowghi, D., 2003. An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations. In: Proceedings of the HICSS'03. IEEE, 10.
Daniel, S., Agarwal, R., Stewart, K.J., 2013. The effects of diversity in global, distributed collectives: a study of open source project success. Inf. Syst. Res. 24, 312–333.
Feller, J., Fitzgerald, B., Hissam, S., Lakhani, K., 2005. Perspectives on Free and Open Source Software. The MIT Press.
Fielding, R.T., 1999. Shared leadership in the Apache project. Commun. ACM 42, 42–43.
Francalanci, C., Merlo, F., 2008. Empirical analysis of the bug fixing process in open source projects. Open Source Dev. Commun. Qual., 187–196.
Grinter, R.E., Herbsleb, J.D., Perry, D.E., 1999. The geography of coordination: dealing with distance in R&D work. In: Proceedings of the GROUP1999. ACM, 306–315.
Guimera, R., Uzzi, B., Spiro, J., Amaral, L.A.N., 2005. Team assembly mechanisms determine collaboration network structure and team performance. Science 308, 697–702.
Harrison, D.A., Klein, K.J., 2007. What's the difference? Diversity constructs as separation, variety, or disparity in organizations. Acad. Manage. Rev. 32, 1199–1228.
Herbsleb, J.D., Mockus, A., 2003. An empirical study of speed and communication in globally distributed software development. IEEE Trans. Software Eng. 29, 481–494.
Hinds, P., Kiesler, S., 2002. Distributed Work. The MIT Press.
Jehn, K.A., Northcraft, G.B., Neale, M.A., 1999. Why differences make a difference: a field study of diversity, conflict and performance in workgroups. Adm. Sci. Q. 44, 741–763.
Jeong, G., Kim, S., Zimmermann, T., 2009. Improving bug triage with bug tossing graphs. In: Proceedings of the ESEC-FSE 09. ACM Press, 111–120.
Joshi, A., Roh, H., 2009. The role of context in work team diversity research: a meta-analytic review. Acad. Manage. J. 52, 599–627.
Kittur, A., Suh, B., Pendleton, B.A., Chi, E.H., 2007. He says, she says: conflict and coordination in Wikipedia. In: Proceedings of the CSCW2007. ACM Press, 453–462.
Ko, A., Chilana, P., 2010. How power users help and hinder open bug reporting. In: Proceedings of the CHI2010. ACM Press, 1665–1674.
Lave, J., Wenger, E., 1991. Situated Learning: Legitimate Peripheral Participation. Cambridge University Press.
Milliken, F.J., Bartel, C.A., Kurtzberg, T.R., Paulus, P., Nijstad, B., 2003. Diversity and creativity in work groups: a dynamic perspective on the affective and cognitive processes that link diversity and performance, Group Creativity: Innovation Through Collaboration. Oxford University Press, New York.
Mockus, A., T Fielding, R.O.Y., Herbsleb, J., D., 2002. Two case studies of open source software development: Apache and Mozilla. ACM Trans. Software Eng. Methodol. 11, 309–346.
Moon, J.Y., Sproull, L., 2000. Essence of Distributed Work: The Case of the Linux Kernel. First Monday, 5.
Olson, G.M., Olson, J.S., 2000. Distance matters. Hum. Comput. Interact. 15, 139–178.
Raymond, E.S., 2001. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly.
Rigby, P., German, D., Storey, M., 2008. Open source software peer review practices: a case study of the apache server. In: Proceedings of the ICSE'08. ACM Press, 541–550.
Rigby, P.C., Storey, M.A., 2011. Understanding broadcast based peer review on open source software projects. In: Proceedings of the ICSE2011. ACM, 541–550.
Roberts, J., Hann, I.L.H., Slaughter, S., 2006. Understanding the motivations, participation and performance of open source software developers: a longitudinal study of the Apache projects. Manage. Sci. 52, 984–999.
Rullani, F., Haefliger, S., 2013. The periphery on stage: the intra-organizational dynamics in online communities of creation. Res. Policy 42, 941–953.
Sandusky, R.J., Gasser, L., 2005. Negotiation and the coordination of information and activity in distributed software problem management. In: Proceedings of the GROUP2005. ACM Press, 187–196.
Shachaf, P., 2008. Cultural diversity and information and communication technology impacts on global virtual teams: an exploratory study. Inf. Manage. 45, 131–142.

Singh, P.V., 2010. The small-world effect: the influence of macro-level properties of developer collaboration networks on open-source project success. ACM Trans. Software Eng. Methodol. (TOSEM) 20, 6.
Star, S.L., 1989. The structure of ill-structured solutions: boundary objects and heterogeneous distributed problem solving. In: HUHNS, M., GASSER, L. (Eds.), Distributed Artificial Intelligence II. Morgan Kauffmann, Menlo Park, CA.
Storey, M.-A., Treude, C., van Deursen, A., Cheng, L.-T., 2010. The impact of social media on software engineering practices and tools. In: Proceedings of the FSE/SDP 2010. ACM, 359–364.
Suchman, L., 1995. Making work visible. Commun. ACM 38, 56–64.
Tan, Y., Mookerjee, V., Singh, P., 2007. Social capital, structural holes and team composition: collaborative networks of the open source software community. In: Proceedings of the ICIS 2007, 155.
Twidale, M.B. Nichols, D.M., 2005. Exploring usability discussions in open source development. In: Proceedings of the HICSS'05. 198c.
Uzzi, B., 1997. Social structure and competition in interfirm networks: the paradox of embeddedness. Adm. Sci. Q., 35–67.
Uzzi, B., Amaral, L.A., Reed-Tsochas, F., 2007. Small world networks and management science research: a review. Eur. Manage. Rev. 4, 77–91.
Van Knippenberg, D., Schippers, M.C., 2007. Work group diversity. Annu. Rev. Psychol. 58, 515–541.
Viégas, F.B., Wattenberg, M., Dave, K., 2004. Studying cooperation and conflict between authors with history flow visualizations. In: Proceedings of the CHI2004. ACM Press, 575–582.
Wang, J., Carroll, J.M., 2011. Behind Linus's law: a preliminary analysis of open source software peer review practices in Mozilla and Python. In: Proceedings of the CTS2011. IEEE, 117–124.
Williams, K.Y., O'Reilly, C.A., 1998. Demography and diversity in organizations: a review of 40 years of research. Res. Org. Behav. 20, 77–140.
Yamauchi, Y., Yokozawa, M., Shinohara, T., Ishida, T., 2000. Collaboration with Lean Media: how open-source software succeeds. In: Proceedings of the CSCW2000. ACM Press, 329–338.



Jing Wang received her Ph.D. degree in Information Sciences and Technology at Pennsylvania State University in 2013. Her research interests include human-computer interaction, computer-supported cooperative work, social computing, online communities, persuasive technologies, and creativity. She has published studies in a variety of domains, such as software development, education, and healthcare.

Patrick C. Shih received the B.S. degree in computer science and engineering from the University of California, Los Angeles, in 2003, the M.S. degree in Information Networking from Carnegie Mellon University in 2005, and the Ph.D. degree in Information and Computer Science from the University of California, Irvine in 2011. In 2012, he joined the College of Information Sciences and Technology at The Pennsylvania State University as a Research Associate. His current research interests include community informatics, healthcare informatics, virtual communities, and educational technologies. Dr. Shih is a member of IEEE and ACM.

John M. Carroll is Distinguished Professor of Information Sciences and Technology at Pennsylvania State University. Recent books include *Learning in Communities* (Springer, 2009), *The Neighborhood in the Internet: Design Research Projects in Community Informatics* (Routledge, 2012), and *Creativity and Rationale: Enhancing Human Experience by Design* (Springer, 2012). Carroll is editor of the *Synthesis Lectures on Human-Centered Informatics*. Carroll received the Rigo Award and CHI Lifetime Achievement Award from ACM, the Goldsmith Award from IEEE. He is a fellow of AAAS, ACM, APS, HFES, and IEEE. In 2012, he received an honorary doctorate in engineering from Universidad Carlos III de Madrid.