Impression Formation in Online Peer Production: Activity Traces and Personal Profiles in GitHub

Jennifer Marlow
Human-Computer Interaction Institute,
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15210
jmarlow@cs.cmu.edu

Laura Dabbish
Human-Computer Interaction Institute & Heinz College
Center for the Future of Work
Carnegie Mellon University
dabbish@cmu.edu

Jim Herbsleb
Institute for Software Research, School of Computer Science & Center for the Future of Work
Carnegie Mellon University
jdh@cs.cmu.edu

ABSTRACT
In this paper we describe a qualitative investigation of impression formation in an online distributed software development community with social media functionality. We find that users in this setting seek out additional information about each other to explore the project space, inform future interactions, and understand the potential future value of a new person. They form impressions around other users' expertise based on history of activity across projects, and successful collaborations with key high status projects in the community. These impressions influence their receptivity to strangers' work contributions.

Author Keywords
Peer production; Collaborative software development; Impression formation; Activity traces

ACM Classification Keywords
H.5.3. Group and Organization interfaces: Computer-supported cooperative work.

INTRODUCTION
Open source software development is an example of commons-based peer production where "uncertainty about the quality of others is the rule rather than the exception" [33]. In this environment, project owners often receive code contributions from previously unknown others. They must routinely form impressions about the expertise, background, and credentials of these unknown contributors. In this research we sought to understand how these impressions form and how they influence receptivity to contributions.

Surprisingly little previous research has examined impression formation in peer production environments. Research on leadership and the promotion process in Wikipedia touches on this issue, highlighting factors that influence selection for management positions [8]. However, these individuals are typically already active project contributors. Impressions of expertise and suitability for management are based on a history of interactions and quality contributions. The question remains: when and how do contributors form "first impressions" of each other? And how do these impressions influence evaluations of contributions?

At the same time, there is a rich history of research on impression formation in offline contexts. This research suggests that we quickly form judgments of strangers' expertise and trustworthiness based on limited cues (such as posture, dress, interaction style etc.). These judgments can be biased and inaccurate in a variety of ways [6], but inevitably impact how we view and interact with the person. In the work context our social judgments change the way we evaluate someone's suitability for a task or their work quality [10].

Online, a completely different set of cues are available. In online social settings like discussion forums or dating sites individuals form impressions of other participants based on cues like screen names, email addresses, or profile pictures [9, 11]. However, in an open online peer production workspace instrumented with social media, there is the potential to know about the entire collaborative world of another user. This increased amount of information may change the way we understand what someone else knows, is good at, or what they are like as a person. Activity traces have the potential to inform how we assess expertise and to shape our interactions.

Recent work by Dabbish et al. [7] suggests that open source developers make an extensive set of social inferences based on activity traces generated by social media connected to the work environment. For example, developers in their sample inferred others' commitment levels based on recency and volume of code commit activity. Their study also suggested these inferences influenced work outcomes. Participants in their sample described using inferences about individual's attributes in evaluating incoming code contributions or locating new knowledge on the site. However, their study provided only a partial picture of the

CSCW '13, February 23–27, 2013, San Antonio, Texas, USA.

types of inferences formed about individuals and their subsequent influence on receptivity to contributions.

Our goal in this work was to develop a more detailed understanding of impression formation in online peer production. We build on the distributed social cognition model [32], which posits that impression formation is an active process influenced by behavior in a network or group. This model shows that when forming impressions of others, individuals engage in an active process that involves the following steps: 1) choosing whether to obtain information about the target; 2) choosing what information is elicited, and 3) interpreting the elicited information to form a person model (an integrated interpretation of what a person is like.) The distributed social cognition model provides general guidelines about how impression formation occurs but doesn't describe what the process looks like in a specific setting. We address the following research questions to understand how distributed social cognition occurs in the online peer production setting:

RQ1: When do people seek out information about unknown others in an online peer production community?

RQ2: What information do they use to form impressions?

RQ3: How do interpersonal impressions influence evaluations of others' contributions?

In order to address these questions, we conducted an interview-based investigation of impression formation in GitHub. We find that users in this setting seek out additional information about each other to explore the project space, inform communication interactions, and understand the potential future value of a newcomer. They form impressions around other users' expertise based on project and code-related cues, which combine with interaction traces to help influence judgments about how to work with new contributors in the context of receiving and accepting pull requests (code contributions.) Our results inform the design of social technology to support online peer production communities engaged in knowledge-based work.

BACKGROUND

Open source projects rely on contributions from a global community of developers to perform various tasks ranging from bug reporting to submitting feature requests and contributing patches and code. The success of a project depends on the proactive and constructive participation of contributors to the project [30]. However, these outside contributions may vary in quality depending on the skills and expertise of the people who contributed them. Diversity in technical abilities can be helpful to a project because various types of contributions, from filing bugs to suggesting features, can be made by people with a range of expertise [30]. However, bug reports can waste time and divert developers' attention when they are misleading due to contributors' inexperience [19]. In both of these cases, the potential benefits of receiving contributions from inexperienced, unknown others may be outweighed when editing their work becomes too time-consuming to deal with, and this often results in the suboptimal outcome of contributions not being accepted to the project [28].

Although we know that individuals engaged in open source software development are continuously evaluating the contributions of others, few studies have addressed exactly when and why developers seek information about unknown others, how impressions of these people are formed, and what information is relied upon to infer an unknown developer's expertise, or other personal characteristics. In the next section, we consider previous work on impression formation and uncertainty reduction to inform our study of this process in the peer production context.

Impression formation and expertise perceptions

Forming impressions of strangers can be thought of as an uncertainty reduction process motivated by the goal of understanding their behavior and predicting how they will behave in the future [11]. When people are faced with a previously unknown person, they can use direct social interaction or information gathering to reduce uncertainty about that person [26]. During the process, the seeker fills out their mental models or mental representations of these new unknown people that help him or her to make sense of other people and their intentions, emotions and behaviors [1]. These can be models about how they will react to certain situations, but also what they know.

Research in sociology and social psychology describes interpersonal impression formation in offline settings. Goffman was among the first to describe the social inference process, by which we interpret characteristics of others based on appearance and public behavior [13]. Later work in social psychology described the cognitive processes associated with mentally placing individuals in social categories [2]. This work showed that we use these broad social categories to populate initially simplistic mental models of an individual (otherwise known as stereotyping) [24]. These initial models can be simplistic and inaccurate, and only through direct experience can we develop accurate impressions of an individual.

Limited research has examined how work-related impressions form through technology. Research in corporate settings has examined how individuals use technology to explore what other people know or are good at. For example, expertise finding is an important task in the corporate domain e.g. [29] and many internal tools have been developed to help people tag their own and others' expertise [25]. Research on impression formation in corporate settings has found that public profile information influences impressions of work-related skills [25, 29]. This previous research does not describe how the impression formation process works. It is also not clear how the presence or absence of organizational boundaries may change this process.

Open source software development is an example of a peer production community fueled by volunteer contributors interacting, via computer-mediated channels, from all over the world. Research conducted in other peer production settings, such as Wikipedia, has examined impression formation in admin promotion decisions. This work suggests editors attend to history of interaction and work on the site in order to make decisions regarding promoting people to admin status. The information that informs these decisions includes evidence of civil interactions with others, social networks and counts/types of edits made [8]. Similarly, Luther et al. [20] found that people desired information about the quality of past work (through peer ratings) as well as soft skills such as the personality of unknown collaborators when deciding who to work with in an online animation creation community.

Impression formation in software development

However, it is unclear to what extent the findings of impression formation in Wikipedia or artistic collaborations extend to evaluating non-managerial participants or to other online peer production communities, partly due to the nature of the domains. In open source development, participants have a wide range of technical abilities and skills, contribute to a project in different ways for different reasons, and may be motivated by career goals or a desire to build reputation and gain peer recognition [18]. Attribution may be less of an issue in open source development than observed in artistic communities [21, 22] in part because projects are viewed as community property and also because systems like GitHub automatically provide a record of a project's origins and contribution history.

Supporting awareness of teammates in distributed software development to improve both task and social outcomes has been an important research area. For example, Trainer et al. [36] suggest that providing visual traces of work interdependencies between team members can influence trust in distributed team members and help people understand whom to ask for assistance. Other work [37] has also pointed to the ways in which tools such as dashboards and activity feeds can help teammates get a sense of the project and plan their tasks. Begel et al. [3] point to the benefits of using social media at various stages in a software team's lifecycle, particularly with respect to coordination and communication. For example, they point to how social media can be used for effortless knowledge sharing or to help groups infer best practices by observing others' work. However, most of these studies have focused on ongoing, organized teams of developers within an organization, unlike the more volunteer-based collaborations that can occur in open source development, where contributors can vary in the length of their involvement and may not share a specific organizational affiliation. Furthermore, in contrast to teams within organizations, developers in open source settings often lack guidance from a management hierarchy, and thus need to self-coordinate and make autonomous decisions based on whatever information is available to them.

Thus, there are gaps in our knowledge about the mechanisms of impression formation in a peer production environment such as open source where participants span organizational boundaries and have extremely heterogeneous backgrounds. Open-source development has traditionally conducted much of the project-based interaction on message boards or via email lists, where it can be difficult to gain insight into who a person is or what they are good at without repeated interactions. In this type of large-scale online peer production community with hundreds or thousands of members, individuals cannot feasibly evaluate in detail every new contributor that accesses a project or submits work. Open source developers may rely heavily on stereotypes as a means of more efficiently assessing people [24].

In this paper we add depth to the previous work by focusing in greater detail on when people seek information about each other, what information they use to make judgments about others, and how they process it. We are specifically interested in the role that social information plays in the collaborative software development process, with a focus on how this information (or lack thereof) influences the openness to contributions.

Social information in the GitHub environment

In order to examine impression formation in peer production, we focused our data collection on GitHub, a software hosting website with over 2.7 million users hosting over 4.5 million repositories as of December 2012 [12]. GitHub's site design integrates social media functionality directly with code management tools. Two unique aspects of the GitHub environment set it apart from other online communities in terms of personal information available about project contributors.

The first unique aspect of GitHub is the presence of a profile for each individual site member. Figure 1 shows a typical user profile on the GitHub site. Profiles on GitHub include: a) biographical data (such as the date they joined the site and optional details about location, employer, etc.), b) a list of their projects in public repositories (including whether they own the project or forked it from another user, the coding languages used, and a histogram of project activity), as well as c) an "activity feed" that displays the most recent actions they have performed on the site (forking projects, watching other users or projects, submitting pull requests, commenting on code, discussing issues, etc.) Finally, the profile also highlights d) the number of people that follow the profile owner as well as the coders and projects that the profile owner has elected to watch.

Similar to many other social networking sites, from the profile users can interact with other users (message them), view content they posted (their code repositories), or view an "activity stream" of their recent actions and behaviors.

Profiles offering this degree of visibility of individual actions are not frequently and easily presented in other peer production environments (e.g. Wikipedia). GitHub is unique in this regard in both the software development and crowd-based work domains.

The second unique aspect of the GitHub environment is that coding is done publicly. Contributors can get involved in pre-existing projects by forking a project, making their changes, and then issuing a pull request to have their change merged back into the main branch of the project. The details of the work done by a contributor on the fork are also visible to the project owner, making it easy to see what a contributor is doing with the project.

The low effort associated with building on others' work is anecdotally associated with attracting more contributors to a project [27]. This accessibility can provide projects with many eager helpers, but project owners still have to evaluate their potential contributions before they are integrated into the master code base. At the same time, all code activity on the site is associated with a user's identity and public profile, meaning a code contribution can be vetted based on a developers entire past history of contribution on the GitHub site.

In our study we aimed to learn exactly when developers searched for more information about each other on the site, how they used this information to form impressions, and how it influenced their evaluation of code contributions.

METHOD

We conducted interviews with 18 GitHub users focusing in detail on how they formed impressions of new people they encountered on the site. Using information obtained through the GitHub API, we identified GitHub users who owned at least one open source project. Potential interviewees who had publicly-displayed e-mail addresses available on their personal profiles were contacted to see if they would like to participate in the study. We did not offer any incentives for participation.

We focused our recruitment on users with leadership positions in popular projects, who managed incoming contributions on a regular basis. These users were more likely to have recent experience handling contributions from strangers they hadn't interacted with before. We contacted GitHub users that owned at least one project with six or more authorized "editors" (people who were authorized to make changes without approval). The six editor cutoff was chosen based on a descriptive analysis of a subset of projects with activity in the 2 weeks prior to February 20, 2011. This analysis showed that projects with greater numbers of editors had more users watching them and more pull requests received per day.

Of the 18 interviewees (17 male), twelve were based in the U.S. and six were located in Europe. This sample was fairly representative of the GitHub community, where around 80% of its users come from North America and Europe [34]. Sixteen interviewees worked as professional software developers or consultants and used GitHub to host personal projects that they worked on in their spare time, while two were PhD students. Most had been members of GitHub for three to four years (based on the "joining dates" on their profiles), with the newest member having joined just under two years ago. Overall, interviewees were active site users and the largest projects they owned had a median number of 59 project watchers, 21 project forks and 451 contributions.

Interviews lasted between 30 and 60 minutes. They were conducted over Skype, with screen sharing enabled so that both interviewer and interviewee could refer to GitHub profiles and pages. The interviews followed a semi-structured format.

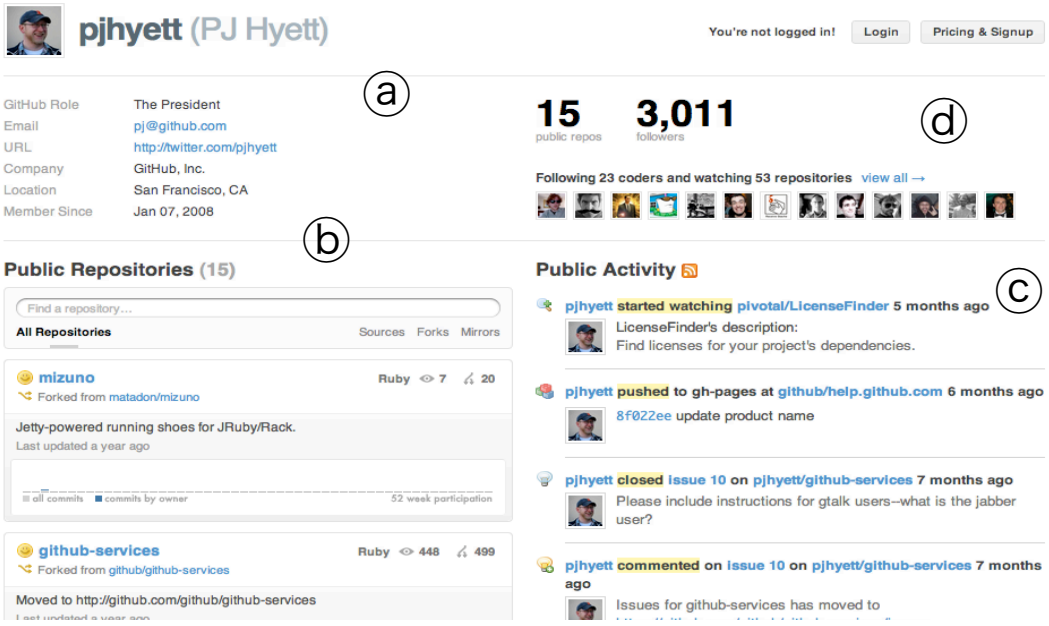

Figure 1. Sample GitHub profile (profile features labeled a-d)

To address RQ1, interviewees were first asked to identify scenarios in which they sought out more information about unknown others. Next, they were asked to go to these people's profiles and describe the information they had focused on and inferences drawn from it. To further address RQ2, each interviewee was asked to think aloud while assessing the profiles of two or three others – some of whom they knew well and others whom they did not know well. Finally, to assess the influence of profiles on work outcomes (RQ3), we also explicitly asked participants to show us examples of pull requests (i.e. code contributions they had received from others) that they had recently dealt with and either accepted or rejected. They described the process of receiving the pull request and walked through how they had handled the request, including whether or not they had consulted the profile of the requester.

Analysis

The analysis process was structured around the three stages in the distributed social cognition model and the corresponding research questions.

Our first research question focused on when people sought personal information about others. To address this question, we began by analyzing interview transcripts for instances in which participants mentioned having consulted others' profiles. Two researchers made affinity diagrams grouping instances around common themes, discussing until consensus was reached, and used these themes to develop categories for impression formation scenarios. We used qualitative analysis software (HyperResearch) to assist in coding interviews and in aggregating similar themes. We then repeated this analysis process for the other research questions.

Our second research question was about which cues individuals used in impression formation. To address this question, we identified all instances of new impressions in the transcripts. We then open coded these instances to form themes around the cues attended to and the impressions resulting from them.

Finally, to address our third research question on the impact of impressions on the working process, we focused on one of the most commonly mentioned themes in the first round of coding: receiving new pull requests from unknown others. We identified pull request interactions described by the interviewees. Next, we generated a set of categories around pull requests in terms of how the request was handled and factors leading to this decision. At some times, discussion was needed to determine the nature of some of the pull request conversations (whether there was a conflict or not) but this was resolved by matching what the interviewees said about the conversation with the correct segments of the visible discussion. Using focused coding [5], we compiled a set of pull requests that demonstrated these themes and then compared and contrasted specific pull requests to create a flow diagram of how decisions to accept pull requests were made, and how exposure to profile elements factored in to this process.

RQ1: SCENARIOS FOR INFORMATION SEEKING

We first investigated when GitHub users sought out personal information in order to learn more about another member. Our analysis identified three scenarios where users sought out more information about each other:

1) discovery, 2) informing interaction, and 3) skill assessment. Each scenario had a different triggering event, questions the user was trying to answer, and the outcome of the search for personal information is summarized in Table 1.

Scenario 1: Using profiles for discovery

Discovery was the first scenario for investigating others. This scenario involved exploring new people's profiles in the hopes of finding interesting projects they could use or learn from. This meant an outcome of discovery was staying up to date with the latest developments in open source world, whether or not these were directly related to the observer's interests and expertise.

Discovery was triggered in two different ways. The first involved encountering an unknown GitHub user with some personal relevance (e.g. the person started watching or forking the interviewee's project or were working on a friend's project). They then looked at the profile of this new person to learn about their other projects in case they were potentially useful. They would browse through the person's projects seeking out common interests, or give projects they had encountered before a second look because the owner had contributed to their project.

The second trigger for discovery was encountering a new project promoted on an external site (Twitter or HackerNews) or on GitHub's 'featured projects' page. Interviewees described becoming aware of a new project through these outlets and then exploring the profile of the project owner to see if there were any other useful projects they could use or learn from (P2, P5, P8, P10, P16). As one interviewee put it:

> *"…they may have done something really interesting, and I'll look at their other projects that they've done and see if any of those are things that I like."* (P8).

Discovery also supported learning about new coding techniques and tools. By checking out someone who had watched or forked their project and delving into what kinds of things they were working on or consulting their personal blogs, people learned about new tools and languages and knew whom they could contact in the future for questions. After viewing the blog of a user who had forked his project, one participant described how this "serendipitous" discovery helped him learn:

> *"[I learned] some very key words about natural language and would I need that at some point, I know where to look and who to contact, so somewhere in my brain, I have this guy connected."* .

Scenario	Trigger	Motivating questions	Outcome
Discovery (mentioned by 10 interviewees)	T1. Receiving a new follower	M1. Curiosity: What cool things does this person do? What is their connection to me?	Find new projects, exposed to new information
Informing interaction (mentioned by 10)	T2. Receiving a pull request	M2. Are they a nice person? Are they worth helping?	Decision of how to act towards the new person based on impressions of interaction style and interests
Forming expectations about skills (mentioned by 6)	T2. Receiving a pull request from unknown person	M3. Who are they? Why are they interested in what I do?	Assessment of what they do for my project, how they can contribute based on skills

Table 1. Scenarios for interpersonal information seeking

Scenario 2: Informing interactions with new people
The second information-seeking scenario we identified focused on informing interaction. This occurred when an unknown person submitted a pull request (offering up a code contribution to the other person's project). Project owners would investigate the profile and activity of the person submitting the pull request to help them decide how to interact with the person (were they receptive to criticism? Did they argue with others?). This was most likely to happen when the project owner had the time and attentional resources to quickly look at the user's profile or when they had encountered the person on more than one occasion.

Because GitHub enables unknown people to fork a project without interacting with the owner, project owners often found themselves receiving pull requests from people they did not know. Owners in our sample were compelled to examine the profiles of these unknown others in conjunction with examining their code to understand why they were interested in the project or submitting a certain change. Learning more about a person informed how they would respond to this person and start building a working relationship: As one interviewee put it, "*I want to know you before I help you"* (P6).

Scenario 3: Forming expectations about skills
The third scenario for seeking information about other people was skill assessment. This involved investigation into a contributor's skills and abilities after they submitted a pull request to a project (P3, P6, P14, P15). Knowing what a person did helped owners to make sense of their abilities or competence as a coder, that is "*who they are and what do they do"*(P3). They looked to see if a person had contributed to other projects to understand in what capacity they might be able to help on their own project (or how much assistance or extra effort accepting their contribution would require, based on their technical abilities.)

Often times this kind of skill assessment did not occur until there had been multiple interactions with a given individual (P10, P12). As one developer explained, repeat contributions to a project made a contributor stand out, and triggered their interest in the face of extreme workload:

> *"for me there's so many different people I interact with on the Internet because my projects are pretty popular so that I just don't have the mental capacity to know each person who I interact with. But let's say if I get another bug from him and then maybe he makes other comments in the future, at some point he'll cross a threshold where I'm like okay, who is this person? What does he or she do?"* (P12).

Dabbish et al. [7] found these repeat contributors were often recruited for more central project involvement, e.g. assigned tasks to complete or bugs to fix.

RQ2: TRACES AND IMPRESSION FORMATION
Our second research question focused on how developers used visible cues in the work environment to form impressions about others. Our analysis identified three categories of impressions formed based on different activity information. Table 2 presents an overview of these categories of impressions, and the types of cues that were used to derive these impressions:

Impression category	Cues
C1. General coding ability	Amount of activity, frequency of commits, number of projects owned vs. forked, length of time on site, languages used
C2. Project-relevant skills	Types of visible activity (coding vs. discussing), specific languages used
C3. Personality and interaction style	Past discussion posts and threads

Table 2. Impression categories and associated cues

General coding ability
Interviewees often made stereotypical judgments about a user's coding ability by quickly scanning the recent activity (or lack thereof) visible on the profile. They mentally categorized other developers as one of three expertise-based personas: newcomers, novices, or competent peers.

Complete newcomers were distinguished by a lack of projects or any activity on their profiles, as well as a recent joining date that corresponded with their contribution. The lack of projects or contributions to other projects made it difficult to infer expertise or intentions. As one interviewee said:

> *"kind of hard to tell how good he is, actually. He hasn't contributed to anything else"* (P14).

Empty profiles also led to conclusions that the person wasn't a committed member of the site or only complained but never coded. One participant mentioned that seeing a person with an empty profile led him to characterize them as a certain "type of person" who signed up to report bugs but didn't actually contribute any code themselves. These people predisposed him to expect a certain type of contribution:

> *"you can usually tell when you go to a profile and they don't have any projects, and they just sign up to communicate with developers, basically. Yeah, it's a bit more frustrating because… sometimes you get bad bug reports or duplicated bug reports, and people who report them are-- I wouldn't say they're-- not negative, but they're-- yeah-- complaining, I would say."* (P13).

Novice or inexperienced users, in contrast, could potentially have started their own projects in addition to forking those of others. However, the kinds of projects that they chose to work on or the types of projects they had started themselves conveyed a lack of expertise. In the former case, one interviewee explained:

> "*Mongo DB is a database that's relatively well known for having a lot of fundamental architectural flaws and performance problems. The fact that he may choose to use this thing, again suggests to me that he's not a very experienced developer."* (P7).

Finally, *competent peers* were judged by the breadth and depth of the projects they owned, and the coding languages they used. As one interviewee described, the languages on a contributor's profile demonstrated 'geek cred':

> "*He seems like a quite talented coder. I mean, I see Pascal, I see Erlang, and yeah, he used VI, so, I mean, he validates his geek cred for all those things. So yeah, that would give me a pretty good first impression on the person, regarding the way he knows how to code"* (P4).

Project-relevant skills

Developers also formed impressions about specific areas of expertise based on a contributor's visible activity. These impressions included the types of work a person was best suited to doing or preferred to do (e.g. did they spend more time writing code or editing and managing projects?)

Activity traces led project owners to form expectations about the ways a person could assist their project. One interviewee highlighted how the portfolio of languages on a user's profile showed what they might be able to contribute to his projects (which were written in C code):

> *"if someone forks and all their other projects are web based PHP stuff, well you can sort of guess that you won't be getting any code patches from them. But if someone writes something and you see that their profile's all really hardcore C libraries, and C stuff, then you can sort of expect them to actually help out and write good C code, for example."* (P3).

Personality judgments

Interviewees also described inferring someone's personality based on how they interacted with others. They used communication activity visible in a person's recent activity feed (such as recent comments a user had made or discussions they had been involved in) to get a sense of what a person was like to work with. Arguments or 'rude' postings revealed negative personality traits such as arrogance or uncooperativeness, or led to the conclusion that someone was difficult to work with. For example, one interviewee talked about inferring willingness to help through exchanges with others:

> *"I think you can kind of look at how they respond in threads, if they're arrogant or if they're trying to be helpful, it does show through a little bit in threads"* (P15).

RQ3: IMPACT OF IMPRESSIONS ON WORK OUTCOMES

Our third research question focused on how these impressions of new contributors affected the working process. In order to address this question, we focused on the pull request evaluation process. Submitting a pull request involves proposing a code contribution to a project which can be accepted, rejected, or commented on and left open. The pull request process is important because it is the primary way open source projects attract new contributors and extend their code base. Our understanding of this process was based on the interviewee's retrospective thinkalouds of recent pull request interactions, which were publicly archived on pages linked to the pull requests.

In the open source software environment, project success is influenced by two key factors, which informed our analysis. First, code submitted to projects should be of high *quality* and not degrade the performance of the project by introducing bugs. It should also be able to be accepted with a minimum amount of effort required from the project owner, because in the open source environment project owners are limited in the amount of their own time they can devote to maintaining their projects – therefore, *efficiency* is valued. Relationship outcomes are also important to project success, because an owner needs to encourage participation from others in order to maintain the vitality of a project. Relationship outcomes suggested by Hackman [14] include *satisfaction* with the relationship, *learning* from distant colleagues, and *desire to work together in the future*. These positive outcomes are most likely to be maximized in the absence of relationship conflict, when developers interact in a polite and cordial manner. In light of these factors influencing open source software project success, we focused in our analysis on the following outcomes associated with a pull request interaction: whether the contribution was accepted, how long the exchange took, and whether conflict arose in the process.

Pull request case studies

We identified ten recent pull requests interviewees had received from contributors they had not directly interacted with before. These pull request interactions varied in terms of whether or not the project owner looked at the profile of the contributor, the nature of the code being submitted, the perceived expertise of the contributor, the amount of discussion surrounding the pull request, and two outcomes:

1) whether or not the pull request was accepted, and 2) the degree to which negative interpersonal interactions were avoided/managed. By comparing and contrasting the ten cases, we gained greater insight into factors that led to a pull request being accepted, and found instances in which impressions based on profile material influenced the attitude of the owner towards the contributor.

Accepted versus rejected pull requests

Our comparison of accepted versus rejected pull requests suggested that uncertainty was a critical factor in the code contribution review process. Owners were more certain about the value of simple changes that addressed features the owner had wanted to add, were small in scope, or fixed a known bug. Owners were less uncertain about the value of code that was suggesting a larger change, introducing a new feature, or conflicting with other existing functionality.

Code that was accepted "as is" was often either straightforward and easily verifiable, or accepted as a matter of principle (A1-A5). Few comments were involved. Declined pull requests, on the other hand, were not immediately accepted although they had the potential to be if certain fixes were made (D1-D5). These requests involved a great deal more uncertainty regarding the implications of accepting the change. These pull requests required some back-and-forth discussion between owner and contributor to explain the reasons why the code couldn't be automatically accepted or would cause problems and then to negotiate a final outcome.

When the value of submitted code was more uncertain, project owners often engaged in an assessment that involved weighing both code-based factors (e.g. perceived value of the code) and person-based factors (e.g. the perceived value of encouraging continued and sustained participation in the project by the new contributor.) For example, the cost of working with someone to fix their code so that it could be accepted (which could be high in cases where contributors were newcomers or novices) was weighed against the potential benefits of helping to mentor a new project member and potentially gain help in the future, or the risks of being annoyed by time-consuming arguments with a novice about why their contribution was not acceptable. Figure 2 summarizes the pull request acceptance decision process.

Impressions influencing mentorship : Examples

In this section we describe two of the five declined pull requests (D3, D4) where the owners reported having examined the profile of the contributor in the process of assessing the code. In both of these cases, the code being submitted was problematic in that it had stylistic issues, followed poor practices, and generally would require additional time and work by the project owner to fix to the point that it could be incorporated into the main branch of the project. However, the impressions they formed from glancing at the contributors' profiles influenced their willingness to accommodate the contributor's efforts.

When the owner of D4 consulted the contributor's profile, he discovered the contributor was a complete newcomer with no previous history. He described his subsequent reaction to this as:

> "*if I see this is their first pull request, I'm more like, 'Oh, thank you. Very nice.' And then try to be more gentle or…more friendly"* (P14).

The results of this impression were that as a result, he shaped his reaction to the code (which followed a poor practice of containing many different commits in one) to be more tolerant. In the end, the code that was submitted later proved to have some problems, so it was not accepted; however, the owner handled the issue in a way that illustrated his desire to encourage rather than sharply criticize someone whom he recognized to be a newcomer.

Similarly, in D3, the project owner received a pull request that was "*a bit problematic in the sense that it gives me a lot of work to accept a patch like this"* (P3). Since his assessment of the contributor, based on profile activity and interactions through the pull request was of "*an intelligent person who has not been coding too long, does not have too much experience in this,"* he realized that this person was *"a good person to focus but needs hand-holding*." For this reason, the owner was willing to look over his work, but rather than outright rejecting it he planned to look it over at a later time.

Impression accuracy: Examples

While we saw evidence of project owners quickly forming impressions of new contributors and using these to frame their interactions with these people, an associated question was to what extent these impressions were accurate. In the six cases where project owners elected to consult the new contributors' profiles, two users had no evidence of activity on their profiles, one was assessed as a novice user, and three were deemed to be competent peers.

In four instances, participants made judgments of others' abilities without having consulted their profiles beforehand (or consulted them after the pull request interaction had occurred.) In three of these four instances, (D1, D2, D5), the default assumption made without looking at the users' profiles was that they were new to GitHub or to coding. The owner of D2 described the contributor as: "*kind of a junior level experience guy, because anyone else would realize that these kind of changes are going to have a dramatic impact on this kind of project if they were more experienced."* (P7).

However, the sender of this pull request was one of our interviewees as well, who had been a professional programmer for over ten years and owned a popular open source project. In this case, had the project owner looked at his profile, he may have formed a different and more positive impression of the contributor's abilities.

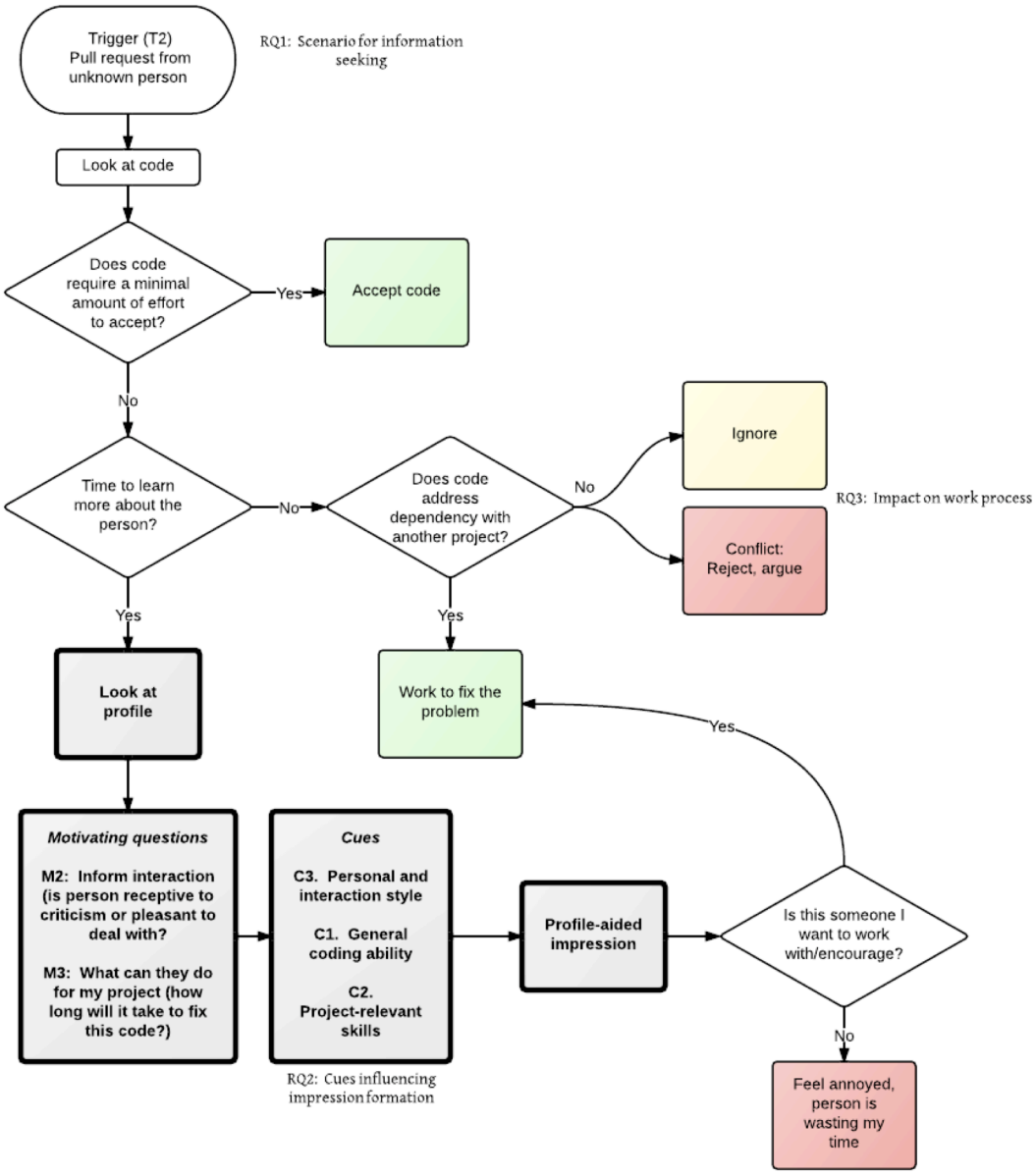

Figure 2. Flow diagram of the pull request decision-making process

DISCUSSION

Our analysis identified three key information-seeking scenarios on GitHub: discovery, informing interaction around contributions, and expertise assessment. Discovery was driven by a desire to stay up to date on new projects, and learn about new people with similar or complementary skills, and did not involve direct interaction.

Developers also sought out more information on others in response to code contributions. They sought out information about other developers' interaction style and interest to inform how they communicated with them. They also formed ability and expertise impressions based on profile information. These impressions influenced the way they handled project contributions moreso when the value of submitted contributions was uncertain, the contributor was unknown and future interactions were not expected. Owners with a stronger tendency for helping would use this information to inform mentoring style interactions to improve a contribution. Bad contributor attitudes combined with low perceived skill and ability led to annoyance, inflexibility, long arguments, and delays.

It is well known that conflict in open source software development often stems from interpersonal matters related to expertise and power. Problems arise when there is mismatched expertise and different knowledge levels [4], newcomers make arguments that are not defended with rationale [17], or people rudely advocate for features that have already been rejected [28]. These types of issues were

seen in many of the example pull request negotiations we elicited. Often, the project owners tended to assume that when unknown contributors sent something for the first time, that they possessed inferior skills to the owner. This corresponds with the assertion made by Oreg & Nov [23] that contributors need to have a reasonable level of expertise and to have this expertise made public in order for them to make a creditable contribution.

The decision of what information to display about users may depend on a site's goals and the behavior it desires to promote (for example, some sites like Wikipedia minimize visible information about contributors in an attempt to create a level playing field [35].) However, when direct cues are absent, it becomes difficult to identify authors and evaluate their expertise. Our work suggests that detailed personal information can shape work outcomes in a peer production setting, particularly for complex contributions.

Design Implications

Like other peer production sites, interactions on GitHub often center around work contributions from unknown others. Our results show that the impressions formed of these users (whether based on activity traces such as profile cues or not) can influence a project owner's receptivity to contributions. In most of the cases where people reported consulting others' profiles, they perceived others as skilled when there was evidence of activity on well-known projects. Project owners may be less receptive to newcomers and developers perceived as unskilled.

Our results suggest that the design of peer production work environments can facilitate more accurate impression formation. First, sites can more accurately portray credentials of a newcomer who may in fact be quite skilled but has not yet built up a portfolio of work on the site. As Luther et al. [20] state, a contributor cannot improve their reputation without establishing an identity and ensuring that the history of contributions is linked to that identity. Finding ways of enhancing cues or links to other credentials or past work may help talented newcomers establish credibility and avoid biased impressions of their skills.

The design of contributor profiles can also help less experienced contributors gain respect from others, particularly given the importance of reputation in peer production environments [18]. A profile cue summarizing relative expertise like Halfaker et al.'s [15] NICE system in Wikipedia, could encourage awareness of and civility towards new editors. For users with some previous activity on GitHub, it could also be useful to show a visual summary of the history of their pull requests submitted to other projects along with an indication of how many of these were accepted or rejected. Such a statistic could help project owners quickly determine whether they will need to spend extra time mentoring or reviewing a person's contributions.

Contributor profiles can also provide better access to communication interaction histories. Our results suggest seeing how someone communicated with others is a useful means of assessing contributor soft skills. This interaction style information may support more effective peer production collaboration across a variety of domains [8, 20]. However, representativeness of interactions displayed is an important design consideration. Activity feeds may bias perceptions of interaction styles since only the most recent behaviors (which may or may not include comments) appear on a person's activity feed on their profile. This means it is possible that an isolated comment may be taken out of context or misconstrued.

Contributor profile design should be optimized for efficiently visualizing or summarizing the information for quick perusal. One of the most frequently mentioned reasons why project owners did *not* consult contributors' profiles was because doing so was simply too time-consuming and inefficient. Because of the site design, they had to scroll through a long list of projects and look at many different icons and cues to determine who owned which projects, what languages they worked in, and so forth. Better organizing this information or summarizing it at the top of the page (for example, a summary of languages and number of projects owned) could reduce the burden of looking at profiles of more experienced members and help more people to quickly assess their expertise and possible contributions.

These design issues around representing peer production contributors can apply to a range of communities. Succinctly summarizing expertise based on behavioral data and incorporating evidence of social interactions may support more nuanced impressions and reduce bias. In any type of peer production site where a person shares their work for others to build on, dealing with contributions from others is necessary and important. Our results show that impressions can influence receptivity to these contributions.

Limitations and future work

In this study we specifically focused on the project owners' assessments of the contributors submitting pull requests. In all cases except for one, it was not feasible to also interview the contributors involved in the pull request exchange to understand the situation from their side (often because they did not provide contact information on their profiles or were located in time zones that made it difficult to find a time for a synchronous interview.) Therefore, our findings do not apply to how contributors form impressions of project owners (however, this perspective has been to some extent covered by [28].)

We also had a limited number of pull request case studies from the interviews that met our criteria of being sent from previously unknown people (many of the other pull requests mentioned in the interviews were from people the interviewees knew and had worked with previously.) In future work, we plan to collect a larger number of instances where the pull request sender is unknown and use these to confirm or modify the model proposed here.

Finally, some of our findings may be unique to GitHub and the OSS community. These communities have their own norms, e.g. "open source" mentality emphasizes mentoring new members in a way that other communities may not share as strongly. Future work should examine the extent to which our results generalize to other open source communities and other peer production settings.

CONCLUSION

We found that open source software developers use detailed traces of an individual's project-related activities for discovery and learning. This information also informs their decisions on how to interact with new, unknown contributors to their projects. Impressions about future potential may increase receptiveness to complex contributions. Our observations suggest that interactions preceded by interpersonal information gathering had more positive outcomes (where social relations were prioritized over efficiency.)

Our results inform the design of future systems to support distributed, computer-supported work. The impression formation process may be expedited by providing more accessible cues about expertise that incorporate activity in other settings. More accurate impressions of expertise and ability may be more critical where there is a perceived gap in skill level between the owner and contributor and the technical merits of the work are unclear and up for debate.

ACKNOWLEDGMENTS

This work was supported by NSF grants IIS-1111750, SMA-1064209, OCI-0943168, a grant from the Center for the Future of Work at Heinz College, Carnegie Mellon University, and an NSF Graduate Research Fellowship.

REFERENCES

1. Antheunis, M. L., Valkenburg, P. M., & Peter, J. Getting acquainted through social network sites: Testing a model of online uncertainty reduction and social attraction. *Comp. in Human Beh.*, *26*(1), (2010), 100–109.
2. Baldwin, M.W. Relational schemas and the processing of social information. *Psychological Bulletin 112*, 3 (1992), 461-484.
3. Begel, A., DeLine, R., & Zimmermann, T. Social media for software engineering. *Proc. FSE/SDP workshop on Future of software engineering research*, (2010), 33–38.
4. Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R., & Zimmermann, T. What makes a good bug report? *Proc. ACM SIGSOFT International Symposium on Foundations of software engineering*, (2008), 308-318.
5. Charmaz, K. Grounded theory as an emergent method. *Handbook of emergent methods*, (2008), 155–170.
6. Chen, S., Shechter, D., and Chaiken, S. Getting at the truth or getting along: Accuracy-versus impression-motivated heuristic and systematic processing. *Journal of Personality and Social Psychology 71*, 2 (1996), 262-275.
7. Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. Social coding in GitHub: transparency and collaboration in an open software repository. *Proc. CSCW*, (2012), 1277–1286.
8. Derthick, K., Tsao, P., Kriplean, T., Borning, A., Zachry, M., & McDonald, D. W. Collaborative sensemaking during admin permission granting in Wikipedia. In *Online Communities and Social Computing* (Vol. 6778), (2011), 100-109).
9. Donath, J.S. Identity and deception in the virtual community. *Communities in cyberspace*, (1999), 29–59.
10. Flynn, F.J., Chatman, J.A., and Spataro, S.E. Getting to know you: The influence of personality on impressions and performance of demographically different people in organizations. *Administrative Science Quarterly 46*, 3 (2001), 414–442.
11. Gibbs, J. L., Ellison, N. B., & Lai, C. H. First comes love, then comes Google: An investigation of uncertainty reduction strategies and self-disclosure in online dating. *Communication Res.*, *38*(1), (2011), 70–100.
12. GitHub.com, https://github.com/home [Accessed December 6, 2012].
13. Goffman, E. The presentation of self in everyday life. New York: Anchor Books, 1959.
14. Hackman, J.R. *Groups That Work (and Those That Don't): Creating Conditions for Effective Teamwork*, San Francisco, CA: Jossey-Bass, 1990.
15. Halfaker, A., Song, B., Stuart, D. A., Kittur, A., & Riedl, J. NICE: Social translucence through UI intervention. *Proc. 7th International Symposium on Wikis and Open Collaboration,* (2011), 101–104.
16. Hamilton, D.L., Katz, L.B., and Leirer, V.O. Cognitive representation of personality impressions: Organizational processes in first impression formation. *Journal of Personality and Social Psychology 39*, 6 (1980), 1050-1063.
17. Ko, A. J., & Chilana, P. K. Design, discussion, and dissent in open bug reports. *Proc.iConference, (*2011), 106-113.
18. Lerner, J., & Tirole, J. Some simple economics of open source. *The Journal of Industrial Economics*, *50*(2), (2002), 197–234.
19. Lotufo, R., Passos, L., & Czarnecki, K. Towards Improving Bug Tracking Systems with Game Mechanisms. GDSLab Technical Report, University of Waterloo, 2011.

20. Luther, K., Caine, K., Ziegler, K., & Bruckman, A. Why it works (when it works): success factors in online creative collaboration. *In Proc. GROUP*, (2010), 1–10.
21. Luther, K., Diakopoulos, N., & Bruckman, A. Edits & credits: Exploring integration and attribution in online creative collaboration. *Ext. Abs. CHI,* (2010), 2823–2832.
22. Monroy-Hernández, A., Hill, B. M., Gonzalez-Rivero, J., & others. Computers can't give credit: how automatic attribution falls short in an online remixing community. *In Proc. CHI*, (2011), 3421–3430.
23. Oreg, S., & Nov, O. Exploring motivations for contributing to open source initiatives: The roles of contribution context and personal values. *Comp. in Human Beh.*, *24*(5), (2008), 2055–2073.
24. Quinn, K. A., Mason, M. F., & Macrae, C. N. Familiarity and person construal: Individuating knowledge moderates the automaticity of category activation. *European Journal of Social Psychology*, *39*(5), (2009), 852–861.
25. Raban, D. R., Danan, A., Ronen, I., & Guy, I. (2012). Impression formation in corporate people tagging. In *Proc. CHI,* (2012), 569–578.
26. Ramirez A., Walther, J. B., Burgoon, J. K., & Sunnafrank, M. Information-Seeking Strategies, Uncertainty, and Computer-Mediated Communication. *Human Communication Res.*, *28*(2), (2002), 213–228.
27. Rao, V. GitHub and the democratization of programming. http://www.forbes.com/sites/venkateshrao/2012/03/27/github-and-the-democratization-of-programming/ [Accessed 18 April 2012]
28. Rigby, P. C., & Storey, M. A. Understanding broadcast based peer review on open source software projects. *Proc. ICSE,* (2011), 541–550.
29. Shami, N. S., Ehrlich, K., Gay, G., & Hancock, J. T. Making sense of strangers' expertise from signals in digital artifacts. *Proc. CHI*, (2009), 69–78.
30. Sinha, V. S., Mani, S., & Sinha, S. Entering the circle of trust: developer initiation as committers in open-source projects. *Proc. of the 8th working conference on Mining software repositories*, (2011), 133–142.
31. Skeels, M., & Grudin, J. When social networks cross boundaries: a case study of workplace use of facebook and linkedin. *Proc. GROUP,* (2009), 95–104.
32. Smith, E. R., & Collins, E. C. Contextualizing person perception: Distributed social cognition. *Psych. Rev.*, *116*(2), (2009), 343-364.
33. Stewart, D. Social status in an open-source community. *American Sociological Review*, *70*(5), (2005), 823–842.
34. Takhteyev, Y., & Hilts, A. Investigating the geography of open source software through GitHub. http://www.takhteyev.org/papers/Takhteyev-Hilts-2010.pdf
35. Tausczik, Y.R. and Pennebaker, J.W. Predicting the perceived quality of online mathematics contributions from users' reputations. *Proc. CHI*, (2011), 1885–1888
36. Trainer, E. H., Al-Ani, B., & Redmiles, D. F. Impact of collaborative traces on trustworthiness. *Proc. of the Workshop on Cooperative and human aspects of software engineering*, (2011), 40–47.
37. Treude, C., & Storey, M. A. Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds. *Software Engineering, 2010 ACM/IEEE 32nd International Conference,* 2010, 365–374).