Impact of Peer Code Review on Peer Impression Formation: A Survey

Amiangshu Bosu, Jeffrey C. Carver,
Department of Computer Science
The University of Alabama
Tuscaloosa, AL 35487-0290
asbosu@ua.edu, carver@cs.ua.edu

***Abstract*—Peer code review has been adopted as an effective quality improvement practice by many Open Source Software (OSS) communities. In addition to increasing software quality, there is anecdotal evidence that peer code review has other benefits, including: sharing knowledge, sharing expertise, sharing development techniques, and most importantly building accurate peer impressions between the code review participants. To further investigate the presence of these benefits, we surveyed members of popular OSS communities who were involved with peer code review. We used established scales from Psychology, Information science, and Organizational Behavior to create survey questions. We also enforced multiple reliability and validity measures to ensure higher confidence in the survey results. In this paper, we present a subset of the surveys results focused on better understanding four aspects of peer impression formation: trust, reliability, perception of expertise, and friendship. The results indicate that there is indeed a high level of trust, reliability, perception of expertise, and friendship between OSS peers who have participated in code review for a period of time. Because code review involves examining someone else's code, unsurprisingly, peer code review helped most in building a perception of expertise between code review partners.**

***Keywords*-Open Source, OSS, Survey, Code Review, Peer impressions, trust, reliability, expertise, friendship**

I. Introduction

The Internet has enabled software development teams to collaborate without the need to be co-located. Developers who never meet face-to-face can collaborate via technologies like email, distributed version control, wikis, instant messaging, video conferencing, and chat rooms. This type of communication and collaboration is often described as Computer-Mediated Communication (CMC) [14]. Despite the availability of various CMC technologies, distributed projects often do not perform as well as co-located projects, presumably because distributed projects require more time and resources to do the same amount of work [16] and are more likely to fail [27].

In distributed teams, where developers do not physically interact, teammates have a difficult time obtaining an accurate perceptions of the expertise and ability of their teammates. These inaccurate perceptions may lead developers to improperly disregard teammates, ultimately reducing the productivity of the virtual team. Because software development productivity can vary by a factor of 10, even across programmers with similar levels of experience [25], we expect that a developer's impressions of his peers significantly impacts his ability to collaborate successfully with those peers and, ultimately, affects project outcomes in distributed software projects.

Open Source Software (OSS) projects are prime examples of distributed software development. Previous empirical research found that OSS participants are motivated by the prospects of enhancing their reputation and by being identified with a particular OSS community [15], [30]. Because peer recognition is a major motivator for OSS participants and has great influence on the success of OSS projects, it is important to understand the peer recognition process within OSS communities.

There is a large body of Psychology research about peer recognition, referred to as "impression formation", from which we can draw. According to the Psychology literature, peer impression is "the judgments that a person, called the perceiver, makes about another person, called the target". The formation of peer impressions primarily depends upon how well the perceiver is acquainted with the target and upon the personality traits of those two individuals [21]. Similarly, Moore defined impression formation as: "the process by which individuals perceive, organize, and ultimately integrate information to form unified and coherent situated impressions of others" [28].

Due to the marked differences between Face-to-Face (FTF) and CMC interactions [1], the impression formation process between OSS participants, who use CMC, is different than the impression formation process between developers working in FTF settings. Because participants who rely on CMC are very task-focused [14], we hypothesize that the sociotechnical interactions (i.e. interactions between people using technologies in workplaces) in which OSS participants can judge their peers' technical contributions (e.g. reviewing code changes, and interacting through bug repositories to resolve bugs) should greatly influence peer impression formation.

Specifically, we hypothesize that peer code review is the sociotechnical interaction in which developers interact most directly and should therefore serve to best support peer impression formation. To test that hypothesis, we conducted a large survey of code review participants in OSS projects. The objective of the survey was to understand: 1) the OSS code review process, 2) expectations of the code review process, and 3) how code review impacts peer impression formation. This paper describes partial results of that survey that focus on the third objective.

The remainder of the paper is organized as following. Section II defines the hypotheses of this study. Section III describes the research method. Section IV describes the results of the reliability and validity analysis of the study. Section V describes the characteristics of the study participants. Section VI discusses the results of the study. Section VII discusses the implications of the study results. Section VIII describes the threats to validate. Finally, Section IX provides some directions for future work and concludes the paper.

II. Research Hypotheses

Software inspections, in which developers subject their code to review by peers or other stakeholders to identify defects, was developed by Michael Fagan in 1976 [12]. Since then, it has been established as an effective quality improvement practice [6], [35]. Even with the benefits offered by software inspections, their relatively high cost and time requirements have reduced the prevalence with which software teams, in general, adopt them [19]. Conversely, many mature, successful OSS projects have recently adopted peer code reviews (informal software inspection) as an important quality assurance mechanism. In this paper we define, **Peer Code Review** as, *the process of analyzing code written by a teammate (i.e. a peer) to judge whether it is of sufficient quality to be integrated into the main project codebase*. There are multiple ways to perform a code review, but they all have a similar goal of evaluating certain properties of newly written or revised code. Besides detecting defects and maintaining the integrity of the code, peer code review helps the community spread knowledge, expertise, and development techniques among the review participants [35].

Our previous motivating work on OSS projects suggests that OSS participants consider a peer's coding contributions to be the most important factor, as opposed to other personal or professional factors, in forming impressions about that peer [2]. Code review is an important method for evaluating the code contributions of one's peers. Moreover, reviewing code may be the most direct interaction that an OSS participant has with his or her peers and therefore a good mechanism to become familiar with the peers' personalities and abilities. Hence, the general hypothesis of the study described in this paper is: *Peer code review is highly effective in building peer impressions between OSS peers.* To study this general hypothesis, the following sections define detailed hypotheses about four important types of peer impression.

A. Trust

We define **Trust** as: *an expectation that someone will act or not act in certain ways.* Participation in the code review process may have both positive and negative impacts on the trustworthiness of an OSS peer. For example, if a code reviewer repeatedly observes that a peer produces high-quality code, he will consider that peer's future code to be trustworthy and will be more likely to trust that peer's project related decisions. Conversely, if a code reviewer finds that a peer writes malicious code, he will likely have difficulty trusting that peer in the future. We seek to determine whether peer code review has an impact (either positive or negative) on trust between OSS participants. The hypotheses related to *Trust* are:

$H1_0$: Participation in peer code review does not affect the mutual trust between OSS peers.

$H1_A$: Participation in peer code review affects the mutual trust between OSS peers.

B. Perception of Expertise

We define **Perception of Expertise** as: *a belief about the amount of various types of expertise a person possesses.* Again, due to the virtual nature of OSS projects, the participants are likely to be less aware of the backgrounds of their peers than participants in FTF projects. OSS participants have to judge the expertise of a peer based primarily upon project contributions. We believe that interactions in which participants can judge their peers contributions will help them to form accurate perception of expertise. The interactions during the code review process tend to focus on the quality and appropriateness of code and should help with forming an accurate perception of expertise. The hypotheses related to *Perception of Expertise* are:

$H2_0$: Participation in peer code review does not affect the perception of expertise between OSS peers.

$H2_A$: Participation in peer code review helps OSS peers form a more accurate perception of expertise of each other.

C. Reliability

We define **Reliability** as: *a belief about the how accurately and completely a teammate will carry out an assigned responsibility.* A code reviewer will judge a peer to be reliable when she finds that peer writing high-quality code, maintaining the project design, following coding guidelines, and adequately testing her code. Conversely, a code reviewer will consider a peer to be unreliable when she writes poor code, violates project guidelines, and introduces bugs. We seek to determine whether peer code review has an impact (either positive or negative) on the reliability with which one views their peers. The hypotheses related to *Reliability* are:

$H3_0$: Participation in peer code review does not affect the perception of reliability between OSS peers.

$H3_A$: Participation in peer code review affects the perception of reliability between OSS peers.

D. Friendship

We use the Oxford English Dictionary definition of **Friendship**: "*a relationship between two or more people who hold mutual affection for each other*" [10]. The OSS participants who are more experienced with regard to the design and constraints for a particular module typically review most of the code changes for that module. It is common for same person(s) to frequently review the code changes from a particular code author. Because the reviewer and the code author will communicate more frequently with each other than with other peers, they may form a friendship. Conversely, code review can be a source of conflict. A code author may consider a reviewer's

rejection or critique of his code to be unfair and become offended. This situation may result in arguments between the code author and the reviewer, and eventually deteriorate their level of friendship. We seek to determine the impact (positive or negative) of peer code review on friendships between OSS participants. The hypotheses related to *Friendship* are:

$H4_0$: Participation in peer code review does not affect the friendship between OSS peers.

$H4_A$: Participation in peer code review affects the friendship between OSS peers.

III. Research Method

To investigate the hypotheses defined in Section II and to better understand the effects of the OSS code review process on peer impression formation, we surveyed code review participants from a number of OSS projects. The following sub-sections describe the participant selection criteria, survey design process, pilot tests, and data collection methods.

A. Participant Selection

Because the impression formation process takes time, peers must have been interacting through code review for a while before impressions can be formed. Many OSS projects use tools (i.e. Gerrit, RietVeld, or ReviewBoard) to manage the code review process. We developed Java applications to populate a MySQL database with data mined from publicly available code review repositories of 34 OSS projects that used one of these code review tools. Then, to identify OSS participants who had been actively involved in code review for a while, we queried the database to find participants who had either posted at least 30 code review requests or reviewed at least 30 code changes. Out of the 12470 code review participants in the database, 2207 (17.8%) matched this criteria.

B. Survey Design

We followed well-regarded social and behavioral research methods to design our survey [9], [13]. For the survey, we defined a *code-review partner* to be "a person who reviews your code or whose code you review on a regular basis" and a *non-code review partner* to be "a person who has been a peer for some time, but you have rarely reviewed their code". Based on these definitions, we designed the survey to determine how OSS participants compare their code review partners to their non-code-review partners on the following topics:

1) trustworthiness,
2) reliability,
3) perception of expertise, and
4) friendship.

We assessed these constructs using multiple questions, most of which were drawn from well-established scales from psychology, information science, or organizational behavior (i.e. *trust* [20], [24], [31], [33], *reliability* [20], [31], [32], *perception of expertise* [32], and *friendship* [3], [32]). We modified and adapted some questions to fit our context. We also created a few new questions based upon our knowledge of OSS development practices. Table I lists the final questions used for the four scales. To avoid any bias, we configured SurveyMonkey to present the questions in a random order and did not include the name of the *construct* with the questions. For each question the survey respondents used a 7-point scale to indicate whether the statement more closely described a code-review partner or a non-code-review partner, with 1 = *describes a code-review partner and NOT a non-code review partner*, 4 = *describes both equally* and 7 = *describes a non-code-review partner and NOT a code-review partner*.

In addition to the questions assessing these four constructs, the survey also contained 21 questions about demographics, the importance of peer code review, and the impact of code quality on peer impressions. In this paper, we analyze the subset of those questions listed in Table II (renumbered for the sake of simplicity).

TABLE I
Constructs and Scale items

Construct	Item	Question
Trust	trust_1	My communication with him/her is more informal (e.g. unofficial, friendly)
	trust_2	S/he is less likely to intentionally misrepresent my point of view to others
	trust_3 (dropped)	I am more likely to consider contributing to a new Free/Open Source project at his/her request
	trust_4	S/he is less likely to take advantage of me (e.g. exploit or deceive for personal benefit)
	trust_5	I am more likely to share my personal information (e.g. feelings, opinions, or achievements) with him/her
Perception of Expertise	expertise1	It is easier for me to identify whether s/he has the ability to provide help in a specific project area
	expertise_2	I more easily know if s/he is the best person to contribute to a specific project area
	expertise_3	It is easier for me to identify if s/he is the right person to fix a given bug report
	expertise_4	I am more aware of his/her level of work and dedication to the project
	expertise_5	I am more likely to seek his/her help in a project-related area (e.g. coding problem, design decision, task assignment, documentation)
Reliability	reliability_1 (dropped)	I am more comfortable assigning a critical task to him or her
	reliability_2	I am more likely to be satisfied with the results of a task assigned to him or her
	reliability_3	S/he is more likely to complete a project-related task (not just code review) s/he accepts even if it requires a large amount of work
	reliability_4	S/he is more likely to follow project coding and design guidelines
	reliability_5	I am more willing to accept his/her advice
Friendship	friendship_1	I would feel a stronger sense of loss at his/her departure from the project
	friendship_2	S/he has a better understanding of me (e.g. aware of my preferences and feelings)
	friendship_3 (dropped)	S/he is more likely to respond to my mailing list posts
	friendship_4	I communicate more frequently with him/her

C. Pilot Tests

To ensure that the survey questions were comprehensible and valid with respect to the study constructs, we conducted multiple pilot tests. First, researchers from Psychology, Computer Science, and Management Information Systems reviewed

TABLE II
SURVEY QUESTIONS

Q1.	Which Free/Open Source project are you most actively involved in?
Q2.	How many years have you worked on Free/Open Source projects?
Q3.	On average, how many people contribute code or review code in a given month for the project?
Q4.	What proportion of the overall code commits in the project undergo peer code review? ◯ Less than 10% ◯ 11% - 25% ◯ 26% - 50% ◯ 51% - 75% ◯ More than 75%
Q5.	Do you receive financial compensation for your participation in the project? ◯ Yes ◯ No
Q6.	Do you think peer code review is important in your project? ◯ Yes ◯ No
Q7.	On the project, how many hours per week, on average, do you spend reviewing other contributors code?
Q8.	From approximately how many different contributors do you review code each week for the project?
Q9.	What is the balance between "code you review" and "code you ask others to review"? ◯ 1- Only a code reviewee ◯ 2 ◯ 3 ◯ 4- Equally a code reviewee and a code reviewer ◯ 5 ◯ 6 ◯ 7- Only a code reviewer
Q10.	What proportion of your code commits do you submit for peer code review? ◯ Less than 10% ◯ 11% - 25% ◯ 26% - 50% ◯ 51% - 75% ◯ More than 75%
Q11.	When you review poorly written code, does it affect your perception of the code author? ◯ Yes ◯ No
Q12.	When you review code of high quality or that has an outstanding approach to solve a problem, does it affect your perception of the code author? ◯ Yes ◯ No

the hypotheses and questions. This review led to several changes, including: changing the rating scales from 5- to 7-point, rewording some questions to remove bias, and adding questions for a broader perspective of the code review process.

Second, software engineering graduate students piloted the survey to identify any difficulties in understanding the questions and to estimate the time required to complete the survey.

Third, 20 OSS code review participants from two projects in our database piloted the survey. This version of the survey had a poor completion rate (i.e. the ratio of number of people staring the survey to the number completing the survey). To address this problem, we rephrased some questions, split the question in which respondents rated the construct scale items (Table I) into two questions, and reordered some questions.

Fourth, we conducted a second pilot with the 24 OSS code review participants from another project in our database. This version of the survey had an adequate completion rate (9 responses). Using these responses, we analyzed the internal consistencies of the four peer impression construct scales. Because the *reliability* scale had questionable internal consistency measures, we added two questions to that construct.

Finally, we conducted a third pilot with 117 OSS code review participants from 11 additional projects. We were satisfied with the completion rate (22 responses) and the scale reliability values. In all pilot studies, we asked respondents to provide feedback and suggestions to improve the survey. We incorporated some of these suggestions into the final survey.

D. Data Collection

On Feb. 14, 2013, we sent a survey invitation to the 2046[1] active code review participants in our database. Removing the 231 emails that were undeliverable, we assume that a total of 1815 code review participants received the survey invitation. On Feb. 26, 2013 we sent a reminder email. We closed the survey on March 10, 2013 after the response rate slowed to almost no responses each day.

Data from the survey link created with Google's URL shortener showed a total 640 clicks on the survey URL (~35% of the invitations). Out of those clicks, 433 people started the survey (~68% of those that clicked on the link) and 287 completed the survey (~66% of those that started). The overall response rate for the survey is ~15.8% (287/1815).

IV. RELIABILITY AND VALIDITY

To reduce uncertainty in survey, it is important to analyze its reliability and validity. The following subsections discuss the reliability and validity of the survey instrument.

A. Reliability

Survey reliability refers to the extent that the survey provides consistent results when replicated in identical conditions [23]. The three most common methods for assessing reliability are: 1) Test-retest, 2) Alternate-form, and 3) Internal consistency. *Internal Consistency* is the most appropriate measure of reliability for this survey.

We used the average inter-item correlation and Cronbach's alpha [8] to test *Internal Consistency*. Chronbach's alpha is widely used to calculate the reliability of multiple-item measurements, with $\alpha \geq .9$ indicating 'excellent', $\alpha \geq .8$ indicating 'good', $\alpha \geq .7$ indicating 'acceptable', and $\alpha < .7$ indicating 'questionable' internal consistency.

The analysis of reliability resulted in dropping three items from the scales to improve the α values: trust_3, friendship_3 and reliability_1. Table III reports the number of remaining items in each scale and the corresponding α coefficients. Two of the scales (*Perception of Expertise*, and *Reliability*) had 'good' internal consistency, and the other two scales (*trust*, and *friendship*) had 'acceptable' internal consistency.

While computing α is essential to analyze the reliability of a survey, other factors are also important. For example, it is possible to achieve a higher α by using more scale items [34]. To combat this problem, we also calculated the inter-item correlations for the scale items, as shown in the last column of Table III. For all four scales, the mean inter-item correlation was $> .4$, which is the suggested minimum average inter-item correlation [4]. Taken together, the α and the inter-item correlation indicate that the survey instrument is reliable.

[1] 2207 total less the 161 used in the pilots

TABLE III
COEFFICIENT ALPHA OF THE SCALES

Construct	Number of items	Cronbach's Alpha	Average inter-item correlation
Trust	4	.756	.438
Perception of Expertise	5	.811	.463
Reliability	4	.810	.518
Friendship	3	.700	.439

B. Validity

Validity of a survey refers to how well it measures what it is intended to measure. This section discuss the analysis of validity relative to the four important types of validity measures for a survey [23].

Face validity: is the weakest form of validity. It focuses on analyzing whether "on its face" each survey item seems like a good operationalization of the construct. We tested face validity by having experts with knowledge of survey design and experts with knowledge of code review in OSS projects review the survey.

Content validity: measures the degree to which the scale items represent the domain of the concept under study. We carefully chose appropriate items from prior validated scales to ensure content validity.

Criterion validity: measures agreement between the survey results and a 'gold standard' criteria (e.g. highly regarded prior survey from same domain). In this case, we could not identify another similar survey to act as a gold standard. Therefore, we were unable to evaluate criterion validity.

Construct validity: measures how well a survey instrument performs in practice. It is the most valuable and most difficult validity measure. Construct validity has two forms:

1) *Convergent validity:* implies that several different methods for obtaining the same information should produce similar result.
2) *Discriminant validity:* implies that measures must not correlate too closely with similar but distinct concepts.

To measure convergent and divergent validity, we performed a principal components analysis with VARIMAX rotation using SPSS. The only restriction we enforced for the factor solution was to retain components with eigenvalue $>$ .8. Table IV summarizes the factor loadings for the four constructs.

The four factors that emerge from the analysis capture 63.1% of the variance (right-most cell in the bottom row of Table IV). As no item loads highly on more than one factor, these four factors are strong. Furthermore, all items load together on the target factor, with the lowest loading being 0.52. According to Comrey, loadings > 0.45 can be considered fair, those > 0.55 as good, those > 0.63 very good, and those > 0.71 as excellent [7]. For the factors in this survey, nine are in the *excellent* range, four in the *very good* range, two in the *good* range, and only one in the *fair* range. These results show that each scale measures the construct it is supposed to measure and is not influenced by other constructs. Therefore, the survey has high construct validity.

TABLE IV
PRINCIPLE COMPONENTS ANALYSIS WITH VARIMAX ROTATION

Item	Factor1 (Reliability)	Factor2 (Expertise)	Factor3 (Trust)	Factor4 (Friendship)
trust_1	-.090	.322	**.660**	.260
trust_2	.372	.144	**.658**	.038
trust_4	.338	.043	**.733**	.041
trust_5	-.048	.141	**.745**	.340
expertise_1	.152	**.817**	.165	.111
expertise_2	.306	**.726**	.227	.106
expertise_3	.078	**.704**	.110	.313
expertise_4	.343	**.568**	.114	.119
expertise_5	.179	**.516**	.052	.486
reliability_2	**.726**	.208	.141	.118
reliability_3	**.740**	.068	.292	.169
reliability_4	**.767**	.189	-.078	.151
reliability_5	**.713**	.311	.133	.111
friendship_1	.241	.408	.002	**.621**
friendship_2	.080	.068	.378	**.758**
friendship_4	.263	.242	.249	**.661**
Total eigenvalue	6.099	1.692	1.444	.871
% of variance	38.120	10.578	9.027	5.446
Cumulative %	38.120	48.698	57.725	63.172

V. DEMOGRAPHICS

To provide an overview of the sample, this section describes the demographics of the OSS projects represented and of the survey respondents. This characterization should help readers properly interpret the applicability of the results.

A. Projects represented

We sent the survey to the code review participants from 23 of the 37 code review repositories in our database (we excluded the 14 code review repositories used in the pilot studies). Table V lists the 45 primary OSS projects represented by the respondents, based on their answers to Q1 (Table II). The number in parenthesis in the 'Multiple occurrences' column represents the number of respondents who listed that project. There are two reasons why this question identified 45 projects when the survey was sent to only 23 projects. First, the project from which we obtained the email address may not be the respondent's primary project. Second, some code review repositories are shared by multiple OSS projects, e.g., the GTK and the ITK project both use the Gerrit repository hosted by Kitware[2].

As an indication of the frequency of code review in the projects represented by the respondents, approximately 83% of the respondents indicated that more than 75% of the code changes in their project undergo peer code review. Therefore, the survey respondents represent projects that are actively using peer code review. This result indicates that the sample does a good job of satisfying the participation selection criterion (defined in Section III-A).

As an indication of the number code review participants, Q3 (Table II) asked respondents to report the average number of people who contributed or reviewed code each month.

[2] http://review.source.kitware.com

TABLE V
RESPONDENTS' PRIMARY OSS PROJECTS

Multiple occurrences				Single occurrences				
Qt Project (36)	MediaWiki (17)	Gromacs(9)	VTK (5)	Coreboot	Go	LLVM	pywikipedia	SystemTap
OpenStack (32)	oVirt (17)	ITK (9)	Gerrit (4)	CloudFoundry	GTK	OmapBoot	Redis	U-Boot
CyanogenMod (28)	Linux kernel (16)	LibreOffice (8)	AOKP (2)	Fedora	ImageVis3D	OmapZoom	Rockbox	vBlob
TYPO3 (20)	Chromium OS (14)	OpenAFS (6)	Couchbase (2)	FlightGear	Jetty	OpenCL	Sage	WebKit
Android (19)	Eclipse (9)	Scilab (5)	Debian (2)	gcc	JGit	Python	Subversion	X.Org

Interestingly, different respondents from the same project provided conflicting estimates. Moreover, a few of the respondents provided unquantifiable responses (i.e.'too many', 'it varies', 'no idea', or 'have not noticed'). Therefore, we had to use alternate approach to determine the number of code review participants. For projects represented by only one respondent, we used the estimate provided by that respondent. For projects represented by two or more repondents that gave conflicting estimates, we computed our own estimate. Because a large percentage of the code is reviewed in most of the OSS projects represented in our sample, the number of code review participants in a month can provide a good estimate of the number of contributors in that month. Using data from the three most recent months, we computed the average number of contributors per month. There were two projects that we did not have adequate data to compute the number of contributors, so we just took the average of the estimates provided by the survey respondents. Using this data, we categorized the projects into four categories:

1) **Small**: $<$ 15 contributors per month (~20% of projects)
2) **Medium**: 16 to 50 contributors per month (~14% of projects).
3) **Large**: 51 to 100 contributors per month (~23% of projects).
4) **Super Large**: $>$ 100 contributors per month (~43% of projects).

These results show that the majority of the projects represented by the sample had a large number of participants in the peer code review process.

B. Respondent demographics

Almost all respondents (99.7%) considered peer code review to be important for their projects. Approximately 90% of the respondents indicated that when code is of high quality or uses and an outstanding problem solving approach, it positively affects their perception of the code author. Conversely, approximately 80% of the respondents indicated that low-quality code negative effects their perception of the code author. Table VI shows the results from survey questions 2, 7 and 8 (Table II). In each case, we grouped the responses into four categories to summarize the data. Figure 1 shows that distribution of answers from question 9 (Table II) regarding the balance between reviewing code and having code reviewed is approximately normal.

Regarding whether the respondents volunteered their time to the OSS project or were paid to contribute, the results showed a 60%/40% split in favor of paid participants. The percentage of paid participants is not surprising because the list of projects we drew from included some large, sponsored projects (e.g. Android, Chromium OS, Qt project, and OpenStack), in which most of the participants are employees of the sponsoring companies. This distribution is in line with previous OSS surveys that had 40%-50% paid OSS participants [2], [22].

TABLE VI
DEMOGRAPHICS OF THE RESPONDENTS

Question	Category description	% of Respondents
2. Experience in OSS projects (Mean= 7 years, $\sigma = 5.3$)	*Low*: Less than 2 years *Medium*: 3 to 5 years *High*: 6 to 10 years *Veteran*: More than 10 years	20% 33% 26% 21%
7. Average number of hours per week spent in reviewing other contributors' code (Mean= 6.4 hours, $\sigma = 7.1$)	*Low*: Less than 2 hours *Medium*: 3 to 5 hours *High*: 6 to 10 hours *Very High*: More than 10 hours	30% 32% 26% 12%
8. Number of contributors' code reviewed each week (Mean= 2.1 peers, σ = .98)	*Small*: Less than 2 peers *Medium*: 3 to 5 peers *High*: 6 to 10 peers *Very High*: More than 10 peers	20% 45% 27% 8%

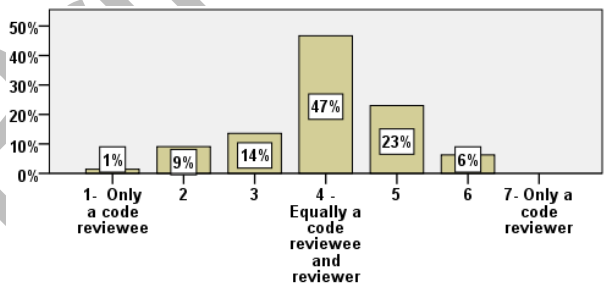

Fig. 1. Balance between being a code reviewer and a code reviewee

VI. RESULTS

Section III-B described the 7-point scale used for the items in Table I. For all analysis in this paper, we recoded the rating scale data inverting the scale and shifting the mid-point to O. The recoded scale is (-3: *describes a non-code review partner, NOT a code review partner*, 0: *describes both equally*, and *3: describes a code review partner, NOT a non-code review partner*). This recoding makes effect of each scale item on peer impression formation more evident because ratings equal to zero mean code review had no effect, negative ratings mean code review had a negative effect and positive ratings mean code review had a positive effect. We did not use this scale during data collection to avoid biasing the respondents with negative scale values.

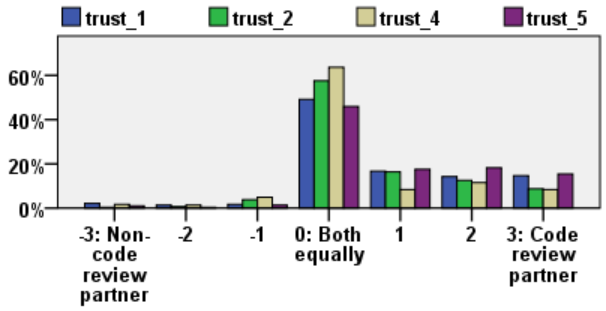

Fig. 2. Ratings distribution for the trust items

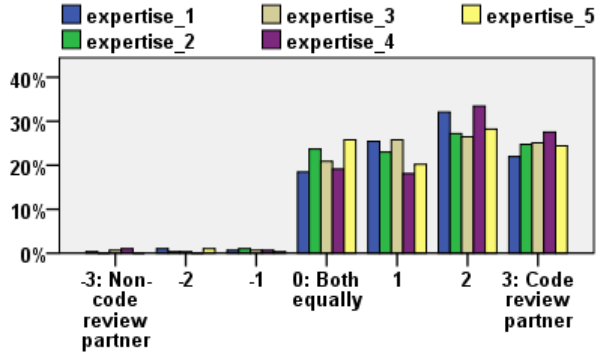

Fig. 3. Ratings distribution for the expertise items

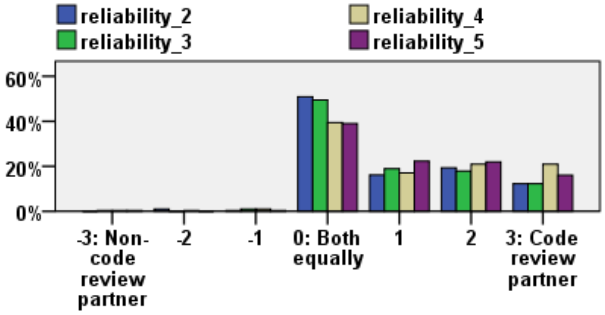

Fig. 4. Ratings distribution for the reliability items

The following subsections describe the effects of code review on the four constructs defined in Section II. For each of the four constructs, we also conducted a One-Way ANOVA to determine whether any of the respondent demographics defined in Section V, i.e. project size, voluntary/paid participation, or experience in OSS projects, affected the results.

A. Trust

The item mean for the four trust scale items (Table I) is .699 with variance of .044. This mean indicates that peer code review has a small but positive effect on trust between OSS code review partners. A one-sample t-test showed that this mean was significantly higher ($t_{283} = 12.93, p < 0.001$) than the mid-point of the scale (*0 - Both Equally*). To determine the impact of this result, we calculated the effect size using Cohen's *d*, with $d \geq .8$ indicating large effect, $d \geq .5$ indicating medium effect, and $d \geq .2$ indicating small effect [5]. The *d* value was .76 indicating a medium-sized effect.

Figure 2 shows that for the four trust scale items, most of the respondents did not think peer code review had an effect on trustworthiness. For those respondents that thought peer code review did have an effect on trustworthiness, most thought it made *code review partners* more trustworthy, resulting in a positive mean value.

Therefore, we conclude that peer code review has a small but significant positive effect on trust between OSS code review partners. We reject the null hypothesis $H1_0$ in favor of the alternate hypothesis $H1_A$.

The results of the One-Way ANOVAs did not show any significant effects of the demographics on the results for the trust construct.

B. Perception of Expertise

The item mean for the five perception of expertise items (Table I) was 1.526 with variance of .004. This mean was significantly higher ($t_{286} = 29.27, p < 0.001$) than the mid-point of the scale (*0 - Both Equally*). The effect size, Cohen's *d* was 1.72, indicating a large effect. Figure 3 shows that for the five perception of expertise scale items, most of the respondents (approximately 70% to 80%) thought they had a better perception of the expertise of their *code review partners* than their *non-code review partners*.

Therefore, we conclude that peer code review has a large, significant effect on the perception of expertise among OSS participants. We reject the null hypothesis $H2_0$ in favor of the alternate hypothesis $H2_A$.

The One-Way ANOVA tests indicated that paid participants had a significantly higher perception of expertise for their *code review partners* than the volunteers participants did (F_1 = 4.58, p =.033). One possible reason for this result could be that paid participants may spend more hours in code review because OSS participation is part of their regular jobs. The results of a Chi-Square analysis showed that, in fact, paid participants do spend significantly more time in code review than volunteer participants do ($\chi^2 = 26.07, df = 3, p < .001$). However, another potential reason for this result could be that paid participants have more opportunities to interact with their peers through FTF mechanisms (e.g. in-person meetings or phone calls) because they are employees of the same company.

C. Reliability

The item mean for the four reliability items (Table I) was 1.0230 with variance of .024. This mean was significantly higher ($t_{282} = 18.55p < 0.001$) than the mid-point of the scale (*0 - Both Equally*). The effect size was large, *d* = 1.11. Figure 4 shows that for the four reliability scale items, about half of the respondents did not think peer code review had an effect on reliability. For those respondents that thought peer code review did have an effect on reliability, most thought it made *code review partners* more reliable, resulting in a positive mean value.

Therefore we conclude that peer code review has a large, significant effect on the reliability of a *code review partner*.

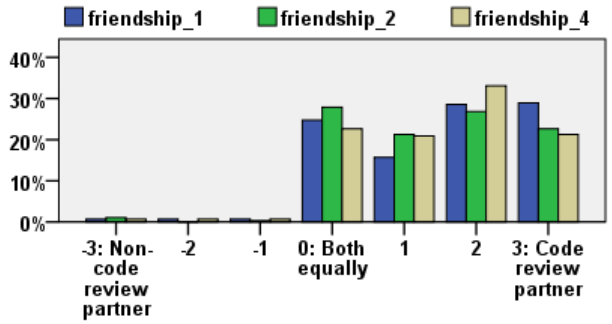

Fig. 5. Ratings distribution for the friendship items

We reject the null hypothesis $H3_0$, and accept the alternate hypothesis $H3_A$.

The results of the One-Way ANOVAs did not show any significant effects of the demographics on the results for the trust construct.

D. Friendship

The item mean for the three friendship scale items (Table I) was 1.472 with variance of .006. This mean was significantly higher ($t_{287} = 25.83, p < 0.001$) than the mid-point of the scale (*0 - Both Equally*). The effect size was large, *d* = 1.53. Figure 5 shows that for the three friendship scale items, most of the respondents (approximately 70% to 75%) thought they were more friendly with their *code review partners* than their *non-code review partners*.

Therefore we conclude that peer code review has a large, significant effect on the friendship between OSS participants. We reject the null hypothesis $H4_0$, and accept the alternate hypothesis $H4_A$.

The One-Way ANOVA tests indicated that paid participants had significantly stronger feelings of friendship towards their *code review partners* than the volunteer participants did. It is possible that the source of this results is the same as for the similar effect on perception of expertise (Section VI-B).

VII. Discussion

The survey results indicate that code review significantly impacts impression formation between OSS participants in four ways. First, code review has the largest effect on forming accurate **perceptions of expertise** between peers. The process of repeatedly reviewing code from a peer allows an OSS participant to understand which areas that peer is contributing to and form an accurate impression about the expertise of that peer. Therefore, we argue that when choosing a peer to fix a bug, or provide assistance in a specific area, it is easier for him to choose appropriately from among his *code-review partners*. Second, code review has a large effect on perception of **reliability** between OSS peers. When a reviewer reviews high-quality code from a peer, she expects her future code to also have good quality. She is also more likely to recommend that peer to others. Third, code review has a large effect on **friendship** betwen OSS peers. Due to increased volume of communications throughout the code review process, the reviewer and the reviewee become more familiar and friendly towards each other. Finally, code review has a moderate effect on the perception of **trustworthiness** between OSS peers. This factor is related to the previous two, because the impression of trustworthiness is built upon producing reliable code and on communicating effectively.

TABLE VII
Correlations between four peer impression constructs

	Trust	**Perception of Expertise**	**Reliability**	**Friendship**
Trust	1			
Perception of Expertise	.472	1		
Reliability	.390	.538	1	
Friendship	.545	.626	.461	1

All correlations are significant at $p < .001$ level

We believed that the four peer impressions constructs might not have been independent of each other. For example, when an OSS participant believes a peer to have high expertise, she may also consider that peer to be reliable on tasks related to her expertise. Conversely, when she loses trust in a peer, here friendship towards that peer may also deteriorate. To better understand these potential relationships, we calculated Pearson's correlation between the four constructs. The results in Table VII indicate all constructs are significantly correlated with each other. Three out of the six correlations are high (i.e $>$ .5). The highest correlation is between **perception of expertise** and **friendship** (*r* = .626). This indicates that the set of peers that OSS participants are friends with has sizable overlap with the set of peer those participants consider to be experts. The next highest correlation is between **trust** and **friendship** (*r* = .545), which is not surprising because trust and friendship are likely to be correlated in any context. Finally, the other high correlation is between **perception of expertise** and **reliability** (*r* = .538), indicating overlap between peers that OSS participants consider to be experts and peers they consider to be reliable.

There is empirical evidence that code review improves software quality [6], [35]. However, the evidence about the other benefits of code review (e.g. sharing knowledge, sharing expertise, sharing development techniques, and most importantly forming peer impressions) has been mostly anecdotal. Our research begins to fill that gap by providing evidence about the effectiveness of code review for peer impression formation. These results also suggest another reason to adopt peer code review.

These results also are applicable to distributed software development (DSD), which is largely similar to OSS development and has been adopted by a large number of companies worldwide [17]. Research suggests that team building process in DSD projects is difficult [18], and that DSD teams often suffer from lack of trust, which negatively affects team performance [26]. DSD organizations use different types of techniques (e.g. photos and profiles of team members at company internal websites, co-located team-building sessions [18], and internal social network sites [11]) to facilitate team-building.

DSD teams suffering from a lack cohesion can adopt code reviews, not only to improve software quality, but also to improve the impressions formation among teammates.

VIII. Threats to Validity

This section discusses the addressed and unaddressed threats to validity in this study. It is organized around the four common types of validity threats.

A. Internal validity

The primary threat to internal validity in this study is *participant selection*. This threat has three aspects that we address as follows.

First, we chose participants based on their level of participation in code review. We used 30 as either the minimum number of code reviews or the minimum number of code changes posted for review. The data indicated that most participants took more than two months to satisfy this criterion. However, it could be argued that two months is not adequate to develop impressions about peers based on code review. While choosing a different threshold could have produced different effects, our results all showed positive effects. So, we do not think that this threat is serious.

Second, despite our efforts to solicit responses from people actively participating in code review, we cannot guarantee that all the respondents met that criterion. There was no motivation for someone who did not fit this criterion to respond to the survey because we did not offer any financial incentives. In addition, we excluded the responses of the respondents who only partially completed the survey. To eliminate any lurkers (i.e. checking the survey questions instead of taking), we manually inspected all open-ended responses searching for random text (a good indication of a lurker). We did not find any lurkers among the respondents who completed the survey. This result indicates a very high quality sample for our survey.

Finally, a survey with a low response rate may be subject to a non-response bias [29]. For example, early respondents might be highly enthusiastic about peer code review and not necessarily representative of all OSS code review participants. Because of the low response rate (~15.8%), we tested for non-response bias. To measure non-response bias, we chose the extrapolation method, which is based on the assumption that respondents who completed the survey slowly were potential non-respondents. In our case, there were two waves of respondents (207 that responded to the initial invitation and 80 that responded to the reminder). We can assume that the second wave of respondents may not have completed the survey without the reminder. To determine whether there was any difference in the responses among the two waves of respondents, we performed independent samples t-tests for the 19 scale items and Chi-Square tests for the 7 multiple choice and demographics questions. There were significant differences for three of the nineteen scale items: trust_4 ($t_{284} = -2.613$, p =.009), friendship_2 ($t_{285} = -2.12$, p=.035), and friendship_4 ($t_{285} = -1.98$, p=.049). To determine whether this non-response bias is a significant threat to our results, we calculated the group differences for the respondents from the two waves. The results in Table VIII show that the scale means were higher for the second wave respondents for all four constructs. Therefore, because the second wave of respondents had higher means (significanly higher in two cases) than the first wave of respondents, the non-response bias is not a serious threat in this study.

TABLE VIII
Group differences in scale mean between two waves of respondents

Construct	First Wave (Mean)	Second Wave (Mean)	Results of t-test
Trust	0.623	0.896	t_{282}= -2.274, p*=.024
Perception of Expertise	1.482	1.640	t_{285}= -1.36, p=.175
Reliability	0.980	1.133	t_{281}= -1.242, p=.215
Friendship	1.391	1.679	t_{285}= -2.282, p*=.023

* indicates a significant difference

B. Construct validity

To reduce the threats to construct validity we took a number of measures. First, we spent approximately eight months carefully designing the survey including expert reviews and multiple pilot tests. Second, we designed the survey itself to reduce bias by:

- placing the questions about the topics of interest after the other survey questions to prevent *hypothesis-guessing* by the respondents,
- configuring SurveyMonkey to present the scale questions in random order,
- providing clear definitions of *code review partner* and *non-code review partner* on all pages containing the scale item questions, and
- carefully wording questions in an unbiased manner.

Third, we conducted multiple reliability and validity tests, with widely-used and highly recommended measures, to ensure construct validity. Therefore, our survey design does not have any significant threats to construct validity.

C. External validity

We cannot definitively establish that our sample is representative of the entire OSS population. OSS communities vary a lot based on product, participant type, community structure, and governance. Because we cannot guarantee that the survey respondents adequately represent the spectrum of OSS participants on these factors, our results may not be applicable to all OSS communities.

Almost two-thirds of the respondents came from large or super large projects (as defined in Section V-A). Moreover, most of the respondents came from well-known, successful OSS projects. These types of projects tend attract more high quality and skilled contributors who tend to produce high-quality code to be reviewed. Therefore, code review may help create more positive impressions about their peers. We may not have observed similar effects if the respondents had come from less successful OSS projects or projects with less governance.

D. Conclusion validity

Conclusion validity is typically threatened by use of inappropriate statistical tests and by drawing conclusions from an inadequate sample size. This survey received an adequate number of responses (287). The analysis used the t-test and ANOVA, which assume normality of underlying sample. In all cases, the skewness and kurtosis were below one indicating that none of the distributions significantly differed from the normal distribution. Finally, the effect size, estimated with Cohen's *d*, was large for three hypotheses and medium for the other hypothesis. Therefore, our study does not have any serious conclusion validity threats .

IX. Conclusions and Future Work

This paper presents the design and results of an online survey of the code review participants in a selection of OSS projects. The contributions of this paper include:

1) Empirical evidence of the effectiveness of code review for peer impression formation between OSS participants,
2) Key insights into the peer impression formation process between OSS participants, and
3) An empirically validated reason for virtual organizations, like OSS projects, managers to adopt peer code review.

Another key contribution of this paper is an illustration of systematically performing a software engineering (SE) survey. Unlike some other more mature fields (e.g., Psychology and Management), most SE surveys lack a systematic evaluation of reliability and validity, leading to uncertainty in the survey results. We believe that to build reliable empirical evidence, SE surveys should be designed systematically and use reliability and validity measures. We hope that our example can help the design of future SE surveys.

In this paper, we report on the results of a subset of the survey questions. There are many open-ended questions that we have yet to analyze in detail. Our future work is to systematically analyze those qualitative responses to gain more insight into code review and peer impression formation in OSS communities. Moreover, using the data already mined from the 34 code review repositories, we plan to perform a comparative study of the code review processes in different OSS projects.

References

[1] J. A. Bargh and K. Y. A. McKenna, "The internet and social life," *Annual Review of Psychology*, vol. 55, no. 1, pp. 573–590, 2004.

[2] A. Bosu, J. Carver, L. Hochstein, and R. Guadagno, "Peer impressions in open source organizations: A survey," Department of Computer Science, University of Alabama, Tech. Rep. SERG-2011-3, 2011. [Online]. Available: http://software.eng.ua.edu/reports/SERG-2011-03

[3] W. M. Bukowski, B. Hoza, and M. Boivin, "Measuring friendship quality during pre-and early adolescence: The development and psychometric properties of the friendship qualities scale," *Journal of Social and Personal Relationships*, vol. 11, no. 3, pp. 471–484, 1994.

[4] L. A. Clark and D. Watson, "Constructing validity: Basic issues in objective scale development," *Psych. Assessment*, vol. 7, pp. 309–319, 1995.

[5] J. Cohen, *Statistical power analysis for the behavioral sciences*. Lawrence Erlbaum, 1988.

[6] J. Cohen, E. Brown, B. DuRette, and S. Teleki, *Best Kept Secrets of Peer Code Review*. Smart Bear, 2006.

[7] A. L. Comrey and H. B. Lee, *A first course in factor analysis*. Lawrence Erlbaum, 1992.

[8] L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, no. 3, pp. 297–334, 1951.

[9] R. F. DeVellis, *Scale development: Theory and applications*. Sage, 2011, vol. 26.

[10] O. E. Dictionary, "Oxford: Oxford university press," 2012.

[11] J. DiMicco, D. R. Millen, W. Geyer, C. Dugan, B. Brownholtz, and M. Muller, "Motivations for social networking at work," in *CSCW*. ACM, 2008, pp. 711–720.

[12] M. E. Fagan, "Design and code inspections to reduce errors in program development," *IBM Systems Journal*, vol. 15, no. 3, pp. 182–211, 1976.

[13] A. Fink, *The survey handbook*. Sage, 2003, vol. 1.

[14] R. E. Guadagno and R. B. Cialdini, "Online persuasion: An examination of gender differences in computer-mediated interpersonal influence," *Group Dynamics: Theory, Research, and Practice*, no. 6, pp. 38–51, 2002.

[15] I.-H. Hann, J. Roberts, and S. Slaughter, "Why developers participate in open source software projects: An empirical investigation," *ICIS*, p. 66, 2004.

[16] J. D. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," *IEEE Trans. on Soft. Eng.*, vol. 29, no. 6, p. 481, june 2003.

[17] J. D. Herbsleb, "Global software engineering: The future of socio-technical coordination," in *2007 Future of Software Engineering*. IEEE Computer Society, 2007, pp. 188–198.

[18] H. Holmstrom, E. Ó. Conchúir, J. Agerfalk, and B. Fitzgerald, "Global software development challenges: A case study on temporal, geographical and socio-cultural distance," in *Int'l Conf. on Global Soft. Eng.* IEEE, 2006, pp. 3–11.

[19] P. M. Johnson, "Reengineering inspection," *Comm. of the ACM*, vol. 41, no. 2, pp. 49–52, 1998.

[20] C. Johnson-George and W. C. Swap, "Measurement of specific interpersonal trust: Construction and validation of a scale to assess trust in a specific other." *J. of Personality and Social Psych.*, vol. 43, no. 6, p. 1306, 1982.

[21] D. A. Kenny, *Interpersonal perception: a social relations analysis*. Guilford Press, 2004.

[22] K. Lakhani, B. Wolf, J. Bates, and C. DiBona, "The boston consulting group hacker survey," *Boston, The Boston Consulting Group*, 2002.

[23] M. S. Litwin, *How to measure survey reliability and validity*. Sage Publications, Incorporated, 1995, vol. 7.

[24] D. J. McAllister, "Affect- and cognition-based trust as foundations for interpersonal cooperation in organizations," *The Academy of Management Journal*, vol. 38, no. 1, pp. pp. 24–59, 1995.

[25] S. McConnell, *What does 10x Mean? Measuring Variations in Programmer Productivity*, ser. Making Software: What Really Works and Why we Believe It. O'Reilly, 2010.

[26] S. P. Mikawa, S. K. Cunnington, and S. A. Gaskins, "Removing barriers to trust in distributed teams: understanding cultural differences and strengthening social ties," in *Proceedings of the 2009 international workshop on Intercultural collaboration*. ACM, 2009, pp. 273–276.

[27] N. Moe and D. Smite, *Understanding Lacking Trust in Global Software Teams: A Multi-case Study*, ser. Product-Focused Software Process Improvement. Springer Berlin / Heidelberg, 2007, vol. 4589, pp. 20–34.

[28] C. Moore, "Impression formation," in *Blackwell Encyclopedia of Psychology*, G. Ritzer, Ed. Blackwell, 2007.

[29] K. Olson, "Survey participation, nonresponse bias, measurement error bias, and total bias," *Public Opinion Quarterly*, vol. 70, no. 5, pp. 737–758, 2006.

[30] E. Raymond, "Homesteading the noosphere," 2001. [Online]. Available: http://catb.org/ esr/writings/homesteading/homesteading/

[31] J. K. Rempel, J. G. Holmes, and M. P. Zanna, "Trust in close relationships." *Journal of personality and social psychology*, vol. 49, no. 1, p. 95, 1985.

[32] J. Robinson, L. Wrightsman, and F. Andrews, *Measures of personality and social psychological attitudes*. Academic Pr, 1991, vol. 1.

[33] J. B. Rotter, "A new scale for the measurement of interpersonal trust1," *Journal of personality*, vol. 35, no. 4, pp. 651–665, 1967.

[34] D. L. Streiner, "Starting at the beginning: an introduction to coefficient alpha and internal consistency," *Journal of personality assessment*, vol. 80, no. 1, pp. 99–103, 2003.

[35] K. E. Wiegers, *Peer reviews in software: A practical guide*. Addison-Wesley Boston, 2002.