TRUST AND DISTRUST IN OPEN SOURCE SOFTWARE DEVELOPMENT

SHUK YING HO
The Australian National University
Canberra, 0200, Australia

ALEX RICHARDSON
The Australian National University
Canberra, 0200, Australia

ABSTRACT

Few open source software (OSS) projects have been great success stories. One reason for this is project stagnation after developers quit their projects. This fact has motivated researchers to examine the factors that influence developers’ intention to continue their participation. One factor is trust among developers. The effects of trust on developers’ intention to remain with their projects have been studied. However, little is known about its conceptual counterpart, distrust. This dearth of knowledge motivates our research. First, we studied what OSS project features affect trust and distrust among developers. Second, we examined how trust and distrust influence developers’ intention to continue participating. We tested our hypotheses with 451 data points from an online survey. Our findings indicate that cooperative norms and effective communication engender trust, whereas an accreditation mechanism eliminates distrust. Additionally, trust positively influences their intention to continue participating, whereas distrust negatively influences it.

KEYWORDS: open source software; trust; distrust; cooperative norms; communication; accreditation; governance.

INTRODUCTION

The Open Source Initiative advocates that open source software (OSS) is built upon the ideals of distributed peer review and process transparency. OSS is characterized as having numerous advantages over closed-source software, such as being highly customizable, promoting localization, and reducing costs through voluntary effort from a dedicated community of developers [3][26]. It is also seen as promoting a fast and cheap development process which emphasizes coding over documentation, and quality over deadlines [9]. Pervasive examples of OSS that have dominated Internet infrastructure include the Apache HTTP Server Project, MySQL and numerous Linux operating system distributions [2]. Disappointingly, most OSS projects halt in the middle of the development, save for a handful of success stories, such as those mentioned above. The failure rate of OSS development reached 50% in 2005, according to the Cyber Security Policy Research Institute [25]. One reason that OSS projects falters is because developers quit the project [13]. When individual developers leave the project, the rest of the team de-energizes and team cohesion declines. This experience has motivated information systems (IS) researchers to examine why developers do or do not continue participating in OSS development [8][13].

Trust among team members has an imperative role in teamwork. By definition, trust is “the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party.” [19] (p. 712). Prior IS research confirms the profound role of how trust among developers influences an individual’s perceptions and/or behavior in the OSS project. For instance, Ljungberg [18] (p. 214) asserted that “[t]he law has been replaced by trust”. Stewart and Gosain [30] developed a model of the OSS community ideology, which predicts how a developer’s trust in other developers influences team effectiveness. Ågerfalk and Fitzgerald [1] revealed a shift from OSS as a community of individual developers, to OSS as a community of small to medium-sized organizations — trust is the key to successfully making this transition. No research, of which we are aware, examines the role of a developer’s distrust in co-developers working on the same OSS project, even though organizational researchers increasingly emphasize distrust [29]. With such a limited understanding of distrust, we cannot represent the complex relationship between trust and distrust operating together to affect OSS outcomes. To address this gap, our first research question is:

RQ1: How does a developer’s trust and distrust in other developers influence his or her intention to continue participating in the OSS project?

This research also looks at OSS features which increase or reduce a developer’s trust and distrust in other developers. Trust arises out of confidence [6]. McAllister [20] argued that there are two bases of trust: cognitive trust is the confidence an individual has in another person’s competence and reliability; affective trust is the confidence an individual has in another person, based on feelings generated by the level of concern and care this person demonstrates. Conversely, distrust is caused by suspicion. It arises when “the disruption of expectations in one exchange is likely to generalize to other” [37] (p. 102). Distrust raises a person’s concern that others may act detrimentally and increases the chance of harmful conflicts. Thus, the distrustful person exists in a state of fear, skepticism, and wariness [17]. Distrust might also lead to sophisticated attributional analyses, which occur when an individual goes to great trouble to actively and carefully scrutinize the potential motives that drive the behavior of others [15]. Taken together, this leads to our second research question:

RQ2: What OSS features influence a developer’s trust and distrust in other developers working on the same project?

This research addresses the above questions through an online survey of 451 OSS developers. From a practical point of view, the findings of the survey reveal how OSS projects can be managed to motivate developers to continue contributing to OSS projects. From a theoretical point of view, this research models developers’
intentions to stay in their projects with trust and distrust, and assesses the combined effects of trust and distrust on the outcome variable.

THEORETICAL BACKGROUND AND HYPOTHESES DEVELOPMENT

Trust and OSS Development

This research looks at two OSS features, cooperative norms and communication, and examines their effects on developers’ trust in co-developers working in the same project. We chose these two variables because they use fundamentally different techniques to build trust. In particular, cooperative norms build trust by altering developers’ ways of thinking. Cooperative norms represent a congruence of understanding about how developers should perform. Effective communication engenders trust through developers’ actions and manifests a positive outcome in developers’ interactions. Examining both variables provides a complete picture of how trust formation is influenced by developers’ thinking and developers’ actions.

Cooperative norms

Norms are the expectations to which a group of people adhere. Cooperative norms are group expectations that emphasize cooperation. Cooperative norms encourage group members to realize that their personal success depends on the success of other developers [36]. Presumably, projects with strong cooperative norms emphasize shared pursuits and mutual interests. Shared pursuits and mutual interests bind people in a project together in collaborative pursuits. This encourages developers to help others. Cooperative norms also encourage people in a project to share information, listen to each other’s ideas, exchange resources, and respond to each other’s requests through positive interdependence [7]. Overall, these interactions reinforce the notion that people in a group have cooperative goals, and they help develop positive expectations of work relationships and work outcome.

Cognitive trust arises from confidence in another person’s competence and reliability [20]. In the context of OSS development, cooperative norms engender cognitive trust for two reasons. First, in a project with strong cooperative norms, developers realize that they are personally successful only when the OSS project succeeds [36]. Thus, they have a common goal; that is, to achieve project success. They work for the sake of the OSS project. Such ideology and collaborative elaboration is important for task completion. When a developer needs help to design a code or fix a bug, other developers are willing to assist. By helping one another, developers demonstrate their skills and knowledge, which develops a positive expectation of competence and reliability. Second, cooperative norms encourage developers to be highly interdependent. They share information and exchange resources, leading to more frequent interactions [6]. Frequent interactions result in rich accumulated knowledge to enable developers to predict, with some level of confidence, the likelihood that their co-developers will adhere to obligations. Taken together, we hypothesize the following:

H1a: Cooperative norms increase a developer’s cognitive trust in other developers working on the same OSS project.

Affective trust, relating to personal emotions, is engendered by feelings generated by the level of concern and care a person demonstrates [20]. Cooperative norms engender affective trust, for two reasons. First, cooperative norms create shared goals. When developers demonstrate a sincere effort to achieve the common goal, they trust the motives of other developers who work in the same project. This leads to a close and secure relationship [30]. Expectations of beneficial conduct from co-developers build affective trust. Second, developers support each other because personal success depends on teamwork. They listen to each other’s ideas, respond to each other’s requests, and provide each other with feedback [7]. People feel that others care about them and are concerned for their welfare. Thus, cooperative norms deepen developers’ relationships, resulting in affective trust. Hence, we hypothesize the following:

H1b: Cooperative norms increase a developer’s affective trust in other developers working on the same OSS project.

Communication

OSS developers are typically geographically dispersed and rarely meet each other. OSS project leaders use different technological and/or managerial approaches to facilitate interpersonal interactions among developers [28]. One common approach is to use electronic communication channels to share resources and solve problems [9]. The community websites that host OSS projects usually set up electronic channels to enable developer-to-project team communication and developer-to-developer communication. For instance, SourceForge provides its OSS developers with blogging, calendar, and forum facilities. These functions help developers organize group events and exchange ideas. They also provide a means by which users can engage in design and socialize with developers and perhaps allow their role to get more involved as a committer [12].

Effective communication engenders a developer’s cognitive trust in other developers for two reasons. First, effective communication means that developers correspond with each other in a frequent and timely manner. This creates a purposeful and organized way for developers to assess the quality of work outcomes. Through communication, developers can also look for ways to improve the work and assure the project’s success. Continuous and timely feedback helps everyone in the project improve their skills and identify program bugs early in the project. This results in developers gaining confidence in the ability of other developers to deliver their assigned program modules. Hence, effective communication helps developers trust the competence and reliability of other developers. Second, effective communication enables developers to understand their co-developers’ motives and increase the level of concern and care among developers. In contrast, when communication is poor, co-developers may not promptly respond to messages, if at all. This creates “virtual silence”, which is a major problem for OSS projects. Virtual silence disrupts workflow and reduces project performance. It creates a situation where developers have to guess how the work is progressing. Therefore, ineffective communication reduces a developer’s cognitive trust in their co-developers because he or she cannot be sure that others are making a reasonable effort to achieve project success. We anticipate that:

Fall 2013
Journal of Computer Information Systems 85

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
H2a: Communication increases a developer’s cognitive trust in other developers working on the same OSS project.

Effective communication engenders a developer’s affective trust in other developers, for two reasons. First, frequent communication allows developers to demonstrate that they care about their co-developers. The amount of concern and care shown indicates intentions and motives. Thus, frequent communication results in goodwill trust [20]. Second, timely and efficient communication maintains and encourages honest behavior. Competent developers who produce high-quality code and/or help others are rewarded by praise, whereas incompetent developers who produce low-quality code are identified by other developers working in the same project. This discourages opportunistic behavior [35]. Since developers expect that others will behave well, this strengthens overall work relationships and builds affective trust among developers. Following this line of reasoning, we propose that:

H2b: Communication increases a developer’s affective trust in other developers working on the same OSS project.

Distrust and OSS Development

Distrust is suspicion that results from discrepancies in individual value systems of people in a team [16]. In our context, value systems refer to developers’ principles of right and wrong that they can do in OSS development. Developers participate in OSS development for different reasons. For instance, some developers want to create better software for the public; some focus on developing software packages for their employers, whereas others look for career and learning opportunities [8]. Their different motives of contribution lead to value incongruence. Thus, developers may suspect each other of not contributing their best efforts to the project, resulting in distrust. This research looks at two OSS features that may reduce distrust: an accreditation mechanism and governance controls. We choose these variables because they are important in the OSS context [28]. An accreditation mechanism operates as a gatekeeper that assesses the capability and/or work attitude of candidate developers before allowing them to join the project, whereas governance controls involve monitoring and verifying the performance of developers currently involved in a project.

An Accreditation Mechanism

A relatively common OSS project management practice is to verify the qualifications of candidate developers before allowing them to join the project [28]. This mechanism is called accreditation. For instance, some projects apply a probation period to newcomers. Candidate developers are allowed to work on a project for a certain period, during which they are closely supervised and their quality is carefully assessed. These candidates may also be current active users that desire an emerging role allowing higher levels of design participation. At the end of the probation, a consensus vote of the core group confers formal membership into the project as a developer [28]. Some projects take a more proactive approach and enthusiastically recruit experts to join the projects. One example is the Linux development project. This project regularly invites legal and software specialists to provide advice, support the Linux brand, and defend litigation [27]. In these situations, accreditation would be high. Conversely, most OSS projects impose minimal accreditation standards. These projects do not have well-defined boundaries. Anyone can simply register to join a mailing list and become a project member [28]. At most, they apply a minimal accreditation effort for some very important positions only. In this situation, accreditation would be low.

Accreditation is a protection mechanism to control the quality of project membership. The accreditation mechanism screens out incapable and less cooperative candidates, so developers who work in accredited projects are likely to believe that all developers are of similar quality — all of them are capable and have a congruent value system. Since developers are all competent and/or have similar value systems, they work for the same goal and this reduces distrust among developers. For instance, a group of altruists works on the project together to develop a software package, and they understand that each contributor wants to develop a software package for the public; and a group of competent developers who work together do not suspect that their co-developers may fail in delivering code modules. This prevents the unpleasant effects of team heterogeneity. Hence, an accreditation mechanism eliminates developers’ suspicions of others’ motives and/or competence. They are unlikely to fear other developers or suspect injurious conduct; thus, there will be little distrust in other developers [17]. Equally, if developers believe that the project’s accreditation mechanism is weak, they may suspect other developers of being incapable, unproductive, and only in the project for selfish gain. These types of suspicions create distrust. Therefore, developers in unaccredited projects are likely to distrust each other more than those in accredited projects. Overall, we anticipate that:

H3: Using an accreditation mechanism reduces a developer’s distrust in other developers working on the same OSS project.

Governance Controls

OSS development is always difficult to manage, for two reasons. First, the membership of OSS developers is very fluid. Developers come from diverse countries and backgrounds [13]. They can freely join the project at any time and leave without notice. Most of them cannot consistently devote large blocks of time to the project [28]. Second, many OSS projects lack a formal organizational structure. The initial developer establishes a project goal, and then calls for volunteers to contribute to the project. Developers participate on an ad hoc basis and have no face-to-face communication with their co-developers. Overall, this means the project has little formal authority, making it difficult to standardize the development process and monitor the progress of each contributing developer.

Governance controls are a set of customs, policies, and institutions affecting the way people direct, administer, or manage a project [23]. They define project expectations, grant power to stakeholders, and verify performance [34]. Governance controls in OSS projects cover four major tasks: managing membership, rules and institutions, monitoring and sanctions, and reputation [28]. Governance controls enable the OSS project leaders to define the roles and accountabilities of developers at the beginning of the project. The project also plans measurable expectations for the desired project outcomes. These measures provide a clear structure
to assure that all activities are conducted according to established expectations for development quality, code delivery, and project performance [28]. Governance also improves coordination in the development phase. During development, a small group of developers measures project output and compares it to measurable expectations. This helps developers understand results achieved and identify possible indications of project failure. Overall, these governance controls monitor and verify project progress. They also reduce uncertainty in the development process and guide the OSS project to completion [34].

In addition, governance controls establish policies and guidelines which clarify relationships and responsibilities among developers. This clarity reinforces awareness of individual responsibility, eliminates role ambiguity, and avoids biased and distorted perceptions. Therefore, governed projects are capable of preventing unwanted outcomes. Moreover, governance controls set up measurable performance indicators and provide a way to assess work progress. They keep track of developers’ obligations to the project by scrutinizing developer activities and chastising wrongdoers. Most developers want to establish a good reputation with their peers, so they behave responsibly. Governance controls reinforce monitoring and verification to ensure that all activities are in accordance with the project’s expectations and specifications. They mitigate uncertainty about malfunctioning activities and unproductive members. Governance controls make information on project activities and work progress available to all developers. This eliminates developers’ distrust that their co-developers are engaging in harmful conflicts or acts. Overall, governance controls align developers’ expectations to narrow their discrepancies whilst monitoring and verification features reduce distrust among developers. Therefore, governed projects display a lower level of distrust than projects without governance.

We anticipate the following hypothesis:

**H4:** Using governance controls reduces a developer’s distrust in other developers working on the same OSS project.

**Effects of Trust and Distrust**

Inasmuch as developers are worried about the competence and motivations of co-developers, trust among developers can help to overcome these concerns. A trusting relationship gives a developer a positive expectation that other developers working in the same project will intent to fulfill their obligations. Trust, associated with hope and assurance [17], also guides the developer to think more optimistically about project outcomes. Thus, the developer does not sense that he or she is being put at risk by the actions of co-developers, even if he or she cannot fully monitor all the activities of the project [35]. Trust also facilitates cohesion and collaboration among developers and leads to more cooperation [29]. These trust effects boost a developer’s intention to remain with the project. Therefore, we anticipate that:

**H5a:** Cognitive trust positively influences a developer’s intention to continue the project.

**H5b:** Affective trust positively influences a developer’s intention to continue the project.

A developer, who distrusts co-developers in the project, suspects co-developers of acting malevolently and concealing their true motives. These suspicions make it difficult to sincerely continue the OSS project. Gradually, the developer becomes defensive and close-minded, weakening team communication [5]. This undermines relationships until no one enjoys the cooperation any more. If undesirable outcomes occur, the developer may blame other developers for neglecting their duty and causing the project to fail. Segmented relationships are likely to occur [17], adding uncertainty to project performance [14]. The developer doubts whether the project will succeed, creating negative expectations about outcomes, and undermining his or her willingness to devote time to the project. Hence, we propose that:

**H5c:** Distrust negatively influences a developer’s intention to continue an OSS project.

**METHODOLOGY**

**Sample**

We conducted an online survey to test the hypothesized relationships. Our target respondents were developers in OSS projects. In 2008, we posted the survey in public OSS discussion forums for eight months. Every month, we posted one or two reminders. The survey was started by 612 developers and we eventually received 451 complete responses. These 451 respondents came from 66 distinct OSS projects. Among the respondents, there were 387 programmers, 18 bug reporters, 16 packagers, 12 testers, and 11 designers. Seven respondents described their position in the OSS project as “Other”. There were 439 males and 12 females. Their average age was 29.

**Survey Instruments**

We measured all research variables using multiple-item scales that were informed by previously validated measures. We adapted items for cooperative norms from Pavlou [24]; items for communication from Stewart and Gosain [30]; and items for accreditation from Pavlou [24]. The construct of governance controls is a formative construct made up of four features, as mentioned in Sharma et al. [28]. The questionnaire items for cognitive and affective trust were adapted from Stewart and Gosain [30]. The items for intention to continue OSS projects were adapted from the theory of reasoned action. The scales were reworded to suit the context of OSS project development. For instance, we combined the wordings from the theory of reasoned action with our OSS context to come up with the items for the intention to continue a project (i.e., “I do not intend to continue this project with my co-developers,” “I predict I would continue this project with my co-developers,” and “I plan to continue this project with my co-developers”).

Distrust was a new construct, because most research on distrust simply negates the items for trust to form the items for distrust. As we could not find a proper instrument to measure distrust, we followed the scales development procedures recommended by Straub [31]. First, we referenced the items used in McKnight et al. [21] and the definitions of distrust used in Sitkin and Roth [29] and Lewicki et al. [17] to develop our own set of items for distrust. Two researchers prepared the draft distrust instrument, which contained ten items. Items for distrust were mixed with items for trust and dissatisfaction to form a questionnaire. Twenty-one undergraduate students from computer science programs filled in the questionnaire and reported ambiguity (if any). We performed
a factor analysis to check the convergent and discriminant validity of the draft instrument, from which we removed five confusing items. For instance, the students found that it was hard to interpret "I suspect that my co-developers have different value systems to me". The second draft instrument had five items and was tested by 13 OSS developers. We performed another factor analysis to confirm the validity of the construct. With the final set of items for distrust, we prepared the whole instrument. Eight postgraduate students pre-tested the whole instrument, which is presented in Appendix 1.

Common method biases may be present in this research because the dependent variable and the independent variables were measured simultaneously and our survey data were self-reported. We conducted a Harman’s one-factor test to assess common method biases. Our results showed five factors accounted for 74% of the variance, with the first factor accounting for 15%. Thus, the threat of common method biases in this research was low. In addition, 14 months after the first survey, we posted a follow-up survey to invite the 451 respondents to report their current participation in OSS projects. Of those, 103 respondents replied to the follow-up survey. A significant correlation (ρ = 0.53, p < 0.01) between a developer’s participation intention (as reported in the first survey) and their actual behavior (as reported in the follow-up survey) confirmed that the threat of common method bias was low in our sample.

**Measurement Validation**

We used partial least square (PLS) to validate our measure and test the model. We first assessed the reliability, discriminant validity, and convergent validity of the reflective constructs. We used internal consistency scores to assess reliability. As shown in Table 1, we calculated composite reliability scores. The internal consistencies of all variables were considered acceptable because they all exceeded 0.90. We used two criteria to assess discriminant validity. First, we checked whether the items loaded much higher on their hypothesized factor than on other factors (own loadings are higher than cross loadings). According to Table 2, we confirmed that all items in reflective items loaded more highly than their cross loadings. Second, we checked whether

**TABLE 1 — Descriptive statistics, correlation matrix, and average variance extracted of principal constructs.**

<table>
<thead>
<tr>
<th>Constructs</th>
<th>Mean (SD)</th>
<th>Reliability</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Cooperative Norm</td>
<td>4.55 (1.62)</td>
<td>0.96</td>
<td>0.87</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Accreditation</td>
<td>4.49 (1.64)</td>
<td>0.94</td>
<td>0.16**</td>
<td>0.86</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Affective Trust</td>
<td>4.58 (1.54)</td>
<td>0.95</td>
<td>0.17**</td>
<td>0.11*</td>
<td>0.87</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Cognitive Trust</td>
<td>4.83 (1.32)</td>
<td>0.93</td>
<td>0.35**</td>
<td>0.13**</td>
<td>0.40**</td>
<td>0.81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Distrust</td>
<td>5.19 (1.35)</td>
<td>0.94</td>
<td>-0.22**</td>
<td>-0.26**</td>
<td>-0.25**</td>
<td>-0.41**</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>6. Intention to Continue a Project</td>
<td>5.26 (1.66)</td>
<td>0.96</td>
<td>0.05</td>
<td>-0.02</td>
<td>0.46*</td>
<td>0.56**</td>
<td>-0.33**</td>
<td>0.92</td>
</tr>
</tbody>
</table>

Notes: * p < 0.05; ** p < 0.01; the diagonal elements (in bold) represent the square root of AVE.

**TABLE 2 — Cross Loadings for Reflective Constructs**

<table>
<thead>
<tr>
<th>COOP</th>
<th>ACC</th>
<th>CTRUST</th>
<th>ATRUST</th>
<th>DTRUST</th>
<th>CONT</th>
</tr>
</thead>
<tbody>
<tr>
<td>COOP1</td>
<td>0.94</td>
<td>0.17</td>
<td>0.20</td>
<td>0.35</td>
<td>-0.21</td>
</tr>
<tr>
<td>COOP2</td>
<td>0.93</td>
<td>0.11</td>
<td>0.16</td>
<td>0.30</td>
<td>-0.19</td>
</tr>
<tr>
<td>COOP3</td>
<td>0.93</td>
<td>0.16</td>
<td>0.15</td>
<td>0.33</td>
<td>-0.24</td>
</tr>
<tr>
<td>COOP4</td>
<td>0.93</td>
<td>0.17</td>
<td>0.15</td>
<td>0.34</td>
<td>-0.21</td>
</tr>
<tr>
<td>COOP5</td>
<td>0.93</td>
<td>0.15</td>
<td>0.16</td>
<td>0.33</td>
<td>-0.19</td>
</tr>
<tr>
<td>ACC1</td>
<td>0.12</td>
<td>0.93</td>
<td>0.10</td>
<td>0.12</td>
<td>-0.23</td>
</tr>
<tr>
<td>ACC2</td>
<td>0.15</td>
<td>0.92</td>
<td>0.13</td>
<td>0.10</td>
<td>-0.24</td>
</tr>
<tr>
<td>ACC3</td>
<td>0.19</td>
<td>0.93</td>
<td>0.10</td>
<td>0.13</td>
<td>-0.25</td>
</tr>
<tr>
<td>CTRUST1</td>
<td>0.17</td>
<td>0.09</td>
<td>0.93</td>
<td>0.39</td>
<td>-0.22</td>
</tr>
<tr>
<td>CTRUST2</td>
<td>0.13</td>
<td>0.13</td>
<td>0.93</td>
<td>0.38</td>
<td>-0.25</td>
</tr>
<tr>
<td>CTRUST3</td>
<td>0.17</td>
<td>0.10</td>
<td>0.94</td>
<td>0.39</td>
<td>-0.25</td>
</tr>
<tr>
<td>CTRUST4</td>
<td>0.17</td>
<td>0.10</td>
<td>0.91</td>
<td>0.35</td>
<td>-0.22</td>
</tr>
<tr>
<td>ATRUST1</td>
<td>0.28</td>
<td>0.11</td>
<td>0.34</td>
<td>0.89</td>
<td>-0.32</td>
</tr>
<tr>
<td>ATRUST2</td>
<td>0.27</td>
<td>0.09</td>
<td>0.39</td>
<td>0.92</td>
<td>-0.37</td>
</tr>
<tr>
<td>ATRUST3</td>
<td>0.41</td>
<td>0.14</td>
<td>0.40</td>
<td>0.92</td>
<td>-0.43</td>
</tr>
<tr>
<td>ATRUST4</td>
<td>0.31</td>
<td>0.12</td>
<td>0.32</td>
<td>0.87</td>
<td>-0.34</td>
</tr>
<tr>
<td>DTRUST1</td>
<td>-0.23</td>
<td>-0.26</td>
<td>-0.32</td>
<td>-0.44</td>
<td>0.92</td>
</tr>
<tr>
<td>DTRUST2</td>
<td>-0.20</td>
<td>-0.25</td>
<td>-0.23</td>
<td>-0.35</td>
<td>0.92</td>
</tr>
<tr>
<td>DTRUST3</td>
<td>-0.17</td>
<td>-0.23</td>
<td>-0.20</td>
<td>-0.30</td>
<td>0.91</td>
</tr>
<tr>
<td>DTRUST4</td>
<td>-0.21</td>
<td>-0.20</td>
<td>-0.12</td>
<td>-0.37</td>
<td>0.85</td>
</tr>
<tr>
<td>CONT1</td>
<td>0.03</td>
<td>-0.01</td>
<td>0.45</td>
<td>0.52</td>
<td>-0.31</td>
</tr>
<tr>
<td>CONT2</td>
<td>0.07</td>
<td>-0.01</td>
<td>0.45</td>
<td>0.54</td>
<td>-0.33</td>
</tr>
<tr>
<td>CONT3</td>
<td>0.03</td>
<td>-0.04</td>
<td>0.43</td>
<td>0.55</td>
<td>-0.31</td>
</tr>
</tbody>
</table>

Note: COOP = Cooperative Norm; ACC = Accreditation. ATRUST = Affective Trust; CTRUST = Cognitive Trust; DTRUST = Distrust; CONT = Intention to Continue an OSS Project
the square root of each factor's average variance extracted (AVE) were larger than its correlations with other factors [4]. As shown in Table 1, the square root of all AVEs was much larger than all other cross correlations. We confirmed that discriminant validity was satisfactory. We used two criteria to assess convergent validity. First, we assessed the AVE for all reflective constructs. Convergent validity was confirmed: the AVEs for all reflective constructs were higher than 0.5. Second, we assessed the significance of item loadings on their reflective factors. As all items were loaded highly in their factors, convergent validity was satisfactory. Together, these findings suggest adequate convergent and discriminant validity.

We then assessed the multicollinearity issue of the formative constructs. This study used two formative constructs: communication and governance. We examined the weights of each item in the constructs. All items were loaded significantly into the corresponding factors. Variance inflation factors (VIF) of items in the governance construct were 3.3, but VIF of items in the communication construct were 4.5. An acceptable value for VIF is 10 [11] with lower values being better. Hence, we find no multicollinearity problem for both constructs.

**FIGURE 1.** Research findings. Note: * p < 0.1, ** p < 0.05, and *** p < 0.01.

**TABLE 3 — Path Coefficients, t-statistics, and p-values**

<table>
<thead>
<tr>
<th>Paths</th>
<th>Coefficient</th>
<th>t-statistics</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1a: Cooperative Norms → Cognitive Trust</td>
<td>0.18</td>
<td>3.61</td>
<td>p &lt; 0.01</td>
</tr>
<tr>
<td>H1b: Cooperative Norms → Affective Trust</td>
<td>0.36</td>
<td>9.42</td>
<td>p &lt; 0.01</td>
</tr>
<tr>
<td>H2a: Communication → Cognitive Trust</td>
<td>0.26</td>
<td>6.52</td>
<td>p &lt; 0.01</td>
</tr>
<tr>
<td>H2b: Communication → Affective Trust</td>
<td>0.53</td>
<td>12.85</td>
<td>p &lt; 0.01</td>
</tr>
<tr>
<td>H3: An Accreditation Mechanism → Distress</td>
<td>-0.26</td>
<td>7.11</td>
<td>p &lt; 0.01</td>
</tr>
<tr>
<td>H4: Governance Controls → Distress</td>
<td>-0.16</td>
<td>1.66</td>
<td>p &lt; 0.1</td>
</tr>
<tr>
<td>H5a: Affective Trust → Intention to Continue Participating</td>
<td>0.27</td>
<td>6.65</td>
<td>p &lt; 0.01</td>
</tr>
<tr>
<td>H5b: Cognitive Trust → Intention to Continue Participating</td>
<td>0.41</td>
<td>7.31</td>
<td>p &lt; 0.01</td>
</tr>
<tr>
<td>H5c: Distress → Intention to Continue Participating</td>
<td>-0.12</td>
<td>1.99</td>
<td>p &lt; 0.05</td>
</tr>
</tbody>
</table>

**FINDINGS**

The PLS Model

Table 3 presents the path coefficients, t-statistics, and p-values for the hypothesized relationships.

The standardized PLS path coefficients are shown in Figure 1. The item loadings of each reflective construct loadings are well above 0.80. As shown in Figure 1 and Table 3, cooperative norms had a direct effect on cognitive trust ($\beta = 0.18$, $t = 3.61$, $p < 0.01$) and affective trust ($\beta = 0.36$, $t = 9.42$, $p < 0.01$), supporting H1a and H1b. As hypothesized, effective communication also had a direct effect on cognitive trust ($\beta = 0.26$, $t = 6.52$, $p < 0.01$) and affective trust ($\beta = 0.53$, $t = 12.85$, $p < 0.01$), supporting H2a and H2b. The R-squares of cognitive trust and affective trust were 11.7% and 42.2% respectively.

Both Figure 1 and Table 3 show that adopting an accreditation mechanism had a significant effect on the reduction of distrust, supporting H3 ($\beta = -0.26$, $t = 7.11$, $p < 0.01$). However, there was no direct effect from governance controls on the reduction of distrust, not supporting H4 ($\beta = -0.16$, $t = 1.66$, $p < 0.1$). The
R-square of distrust was 10.1%. Cognitive trust ($\beta = 0.27$, $t = 6.65$, $p < 0.01$), affective trust ($\beta = 0.41$, $t = 7.31$, $p < 0.01$), and distrust ($\beta = -0.12$, $t = 1.99$, $p < 0.05$) were found to be significant in influencing intention to continue an OSS project, supporting H5a, H5b, and H5c. The R-square of the intention to continue the current OSS project was 38.7%.

**Post Hoc Data Analysis on the Simultaneous Formation of Trust and Distrust**

Lewicki et al. [17] argue that trust and distrust co-exist. In our context, it means that a developer can simultaneously trust and distrust other developers working on the same project. For instance, a developer may trust other developers’ competence, but at the same time, he or she may distrust other developers’ work motivations. According to Lewicki and his colleagues, the four combinations — high trust and high distrust, high trust and low distrust, low trust and high distrust, or low trust and low distrust — are all possible. We performed a post-hoc data analysis to examine the distribution of our respondents in these four combinations. Specifically, our survey asked respondents to report their cognitive/affective trust and distrust in co-developers. We computed the composite scores of affective trust and cognitive distrust for each developer, and then calculated the average of the two to form the score of trust. We also computed the composite score of distrust for each developer. With the cores of trust and distrust for each developer, we calculated the median of trust and the median of distrust for all 451 respondents. In our data, the median of a developer’s trust was 4.89, and the median a developer’s distrust was 5.23. We then performed two median splits — one median split was to divide respondents into high-trust and low-trust groups, and the other median split was to divide respondents into high-distrust and low-distrust groups. In doing so, each respondent had two group labels — each of them was assigned to either high- or low-trust group, and either high- or low-distrust group.

The two median splits classified the respondents into four groups, corresponding to the four combinations specified by Lewicki et al. [17]. Within these four groups, we found that 278 (61.64%) of the respondents belonged to the high trust-low distrust group or the low trust-high distrust group. That is, more than half of the respondents were in a “balanced” psychological state. We then look at the “imbalanced” psychological states, which appear contradictory and uncommon — on the surface. In our sample, however, a significant proportion of trust and distrust combinations existed in the imbalanced psychological states; that is 107 (23.73%) in the low trust-low distrust group and 66 (14.63%) in the high trust-high distrust group. Under the condition of low trust and low distrust, people are neither confident nor suspicious of each other. This psychological state occurs when the degree of interdependence is low. Under the condition of high trust-high distrust, people have shared as well as separate goals. This psychological state occurs when the degree of interdependence is high; and developers trust and verify each other.

1. For each respondent, we calculated a score of trust by averaging the two latent variables, cognitive and affective trust. We sorted the scores and located the median. Then we split the sample into high trust and low trust groups.

**DISCUSSION**

OSS has already proven itself as a potent force behind Internet infrastructure. Many organizations view using OSS as an effective way to reduce vendor lock-in and total cost of ownership, while increasing security through transparency and open peer review [10]. Linux, MySQL, PHP, Apache and other projects are often touted as emminent figureheads for OSS success. Unfortunately, such success stories are greatly outnumbered by OSS projects that have lost momentum and become inactive before their full potential was realized. Nuvolari and Rullan [22] argue that OSS development is dissipative because of the disproportion between inactive OSS projects and successful ones. Subramaniam et al. [32] identified that OSS project activity levels and developers' interest, in a given period, significantly affect measures of project success in the following time period. It is therefore vital to understand how to encourage a developer to continue participating in an OSS project.

This research is one of a few pioneering studies to examine how OSS features influence a developer’s trust and distrust in other developers within the same project. We describe two groups of OSS features — one is for building trust (cooperative norms and communication) whereas the other is for eliminating distrust (using an accreditation mechanism and governance controls) — and examine their effects on a developer’s trust and distrust in other developers. Cooperative norms (H1a, H1b) and communication (H2a, H2b) have a direct effect on a developer’s affective and cognitive trust in other developers working on the same project. The findings are in line with prior research (e.g., [30]). H3 reveals that if developers believe the OSS project spends effort on implementing an accreditation mechanism, they will not distrust their co-developers. The result for H4 is contrary to our predictions. There is only a marginally significant effect of governance on distrust. To probe this result further, we divided our sample into two groups, small projects and large projects, and studied the path coefficients for different team sizes. The small projects group consisted of respondents who had recently worked in an OSS project with fewer than 100 members. The large projects group consisted of respondents who had recently worked in an OSS project with 100 or more members. The result of a multi-group comparison in PLS shows that governance exerted a significant effect on distrust only in the large projects group ($\beta = -0.22$, $t = 1.76$, $p < 0.05$), but not in the small projects group ($\beta = -0.11$, $t = 0.65$, $p > 0.1$). Projects with fewer than 100 members may be capable of coordinating themselves without formal intervention. In other words, only developers in large projects distrust their co-developers, given that other factors are fixed. Our findings imply that distrust is affected by the number of developers in an OSS project.

Additionally, our findings show that whether developers’ participation continues depends on the trusting relationships of other developers within development projects. Specifically, our findings encourage OSS project leaders to focus on cultivating cooperative norms and strengthening communication in their projects to engender trust, which in turn, positively affects developers’ intentions to continue the current projects. If co-developers are helpful and provide useful feedback, developers will have confidence in each other and trust will be engendered. These findings are encouraging. The prolific adoption of Web 2.0 applications has the potential to further become a technological lever in enabling user-centric information-sharing and feedback. Web 2.0 applications typically allow real-time communication,
efficient searching and instant notification of content updates. These features facilitate communication among developers and help cultivate trusting relationships. Voice over IP, instant messaging, group support systems, and video chat also provide means by which to bridge the geographical distance between developers. Without such rich-media mechanisms, communication would suffer, which further hinders project development and success. Creators of popular code repositories for OSS, like SourceForge, have realized the capability of Web 2.0 technology to empower collaboration through web-based software project management tools, wikis, forums and blogs. In fact, many popular Web 2.0 implementations are the result of OSS projects. Through this enablement, important intrinsic motives common to developers, such as sharing and learning [33], are able to flourish.

Lastly, our findings indicate that developers’ participation is hindered if developers distrust each other. To reduce distrust, project leaders should implement an accreditation mechanism to assess the quality of new developers, and project leaders for large OSS projects should impose governance controls to guide and monitor developers’ behavior. These measures reduce distrust among developers, and subsequently, strengthen their intention to continue participating in the projects. In practice, interestingly, there is an inherent trade-off in trying to control “workers” in the OSS context. In OSS development, developers are attracted to an environment that is free of the restrictions normally imposed by corporate software development houses, yet they still want some controls over their co-developers. Nonetheless, controls and compulsory checks give innocent developers a bad feeling, and worse, reduce their intrinsic motivation. It is not possible to attain the best of both worlds. Assuming it is worth investing in these OSS features, project leaders are left to determine how to weigh the competing values of openness and control.

CONCLUSION

In sum, this research has investigated the effects of OSS features that influence developers’ intentions to remain with the OSS projects. The theoretical frame is a developer’s trust and distrust in other developers working in the same project. In general, cooperative norms and effective communication lead to developers’ cognitive and affective trust in other developers. These two kinds of trust favorably affect developers’ intentions to stay in the OSS projects. Conversely, an accreditation mechanism helps reduce distrust, which exerts an adverse effect on developers’ intention to continue their participation. To our surprise, implementing governance controls is not salient in influencing developers’ distrust of their co-developers. Further analyses show that the relationship between governance and distrust is sensitive to the number of developers in the project. Our results provide the impetus to develop a new line of research examining which OSS features moderate the formation of distrust.

This research represents a first step toward understanding how the features of OSS projects affect developers’ trust and distrust in other developers working in the same project, and ultimately their intention to continue the project. It sheds light on the extant literature by examining the interplay between trust and distrust. We also gain an improved understanding of how an OSS project should be structured to retain developers. This is critical to OSS development because our findings help foster trusting relationships among developers and motivate more developers to continue their participation in the OSS projects.

REFERENCES


APPENDIX 1 — QUESTIONNAIRE ITEMS

Cooperative Norms
1. The team encourages developers to work together cooperatively.
2. The team promotes norms of mutual support for developers to resolve any disputes.
3. Developers on the team are willing to make cooperative adjustments to implement the development process successfully.
4. Developers on the team exchange a considerable amount of information throughout the project.
5. Developers on the team discuss problems and exchange feedback throughout the project.

Communication [Formative]
6. Developers answer each other’s questions in a timely manner.
7. Developers’ responses to each other’s questions are correct.
8. Developers’ responses to each other’s questions are useful.
9. Developers answer each other’s questions in a thoughtful manner.

An Accreditation Mechanism
10. I believe that the team undertakes a thorough screening process before people are allowed to join the team.
11. I believe the team makes a substantial effort to assess newcomers’ true competencies.
12. I believe that the team assesses the competencies of newcomers seriously in the selection process.

Governance Controls [Formative]
13. An effective authority is available to verify developers’ performance.
14. An effective authority is available to help resolve conflicts.
15. An effective mechanism is available to assure that all work is in accordance with the posted specifications.
16. An effective mechanism is available to assure that all activities are conducted properly.
Cognitive Trust
17. Developers of the team know that everyone on the team approaches their work with professionalism.
18. Developers of the team know that everyone on the team approaches their work with dedication.
19. Given the past performance records of developers on the team, I do not have confidence in each other's competence and preparation for a job. [a reverse-scored item]
20. Developers of the team believe they will be able to rely on other developers not to make a job more difficult by careless work.

Affective Trust
21. Developers of the team have not made considerable emotional investments in our working relationships. [a reverse-scored item]
22. Developers of the team have a sharing relationship with me. We can freely share our ideas, feelings and hopes.
23. On this team, we can talk freely with each other about difficulties we are having and know that others will want to listen.
24. Developers of the team would feel a sense of loss we could no longer work together.

Distrust
25. I suspect that developers of the team are sometimes out for their own good.
26. I suspect that developers of the team sometimes pretend to care more about others than they really do.
27. I suspect that developers of the team inwardly dislike putting themselves out to help other people.
28. I suspect that developers of the team would tell a lie if they would gain by it.

Intention to Continue a Project
29. I do not intend to continue this project with my co-developers. [a reverse-scored item]
30. I predict I would continue this project with my co-developers.
31. I plan to continue this project with my co-developers.