Transactive Memory System, Communication Quality, and Knowledge Sharing in Distributed Teams: An Empirical Examination in Open Source Software Project Teams

Completed Research Paper

Xiaogang Chen
Southwestern University of Finance and Economics, China
No. 555, LiuTai Ave, Wenjiang District, Chengdu, China, 611130
chenxg@swufe.edu.cn

Abstract

This study develops a research model explaining knowledge sharing among open source software (OSS) developers based on the theories of transactive memory system (TMS) and compensatory adaptation. Specifically, this study proposes that the TMS should be positively associated with knowledge sharing and the quality of communication among OSS developers. The quality of communication is positively associated with knowledge sharing. Furthermore, this study suggests that communication quality partially mediate the positive relationship between TMS and knowledge sharing. By collecting data from 120 OSS project teams, this study empirically confirms all the hypotheses.

Keywords: Open source software, transactive memory system, communication quality, knowledge sharing

Introduction

The open source software (OSS) development differs significantly from traditional in-house software development in that its major workforce is voluntary developers, who are geographically dispersed and rarely, if ever, meet face-to-face (Feller and Fitzgerald 2002). The developers have to rely on Internet tools, such as mailing lists and concurrent version system (CVS), to communicate and collaborate with each other (Wayner 2000). Therefore, an OSS team is viewed as a form of distributed teams with pure virtualness (Griffith et al. 2003; Stewart and Gosain 2006). The pure virtualness poses difficult knowledge management challenges for OSS developers (Crowston et al. 2012), even though they are highly motivated by altruism, learning purpose, career advancement, and personal software need (Bonaccorsi and Rossi 2003; Lerner and Tirole 2002; Raymond 2001; Ye and Kishida 2003).

Knowledge is the most important resource for software development (Robillard 1999), which involves a wide range of specialized knowledge, such as system design, programming, and business rules (Rus and Lindvall 2002). Therefore, effective knowledge management is the key for the success of software development teams. This is true for both traditional software and OSS development. OSS development faces particular knowledge management challenges because of its pure virtualness (Crowston et al. 2012).
Nonetheless, many OSS teams cope with the challenges well since a large number of OSS with good quality has been successfully produced (Stamelos et al. 2002). This interesting phenomenon has drawn attention from the IS community. A number of researchers (e.g., Ciborra and Andreu 2001; Dafermos 2005; Hemetsberger and Reinhardt 2006; Lanzara and Morner 2004; Lee and Cole 2003) have looked into the mechanisms that help OSS teams overcome knowledge management barriers. In brief, these researchers reveal that copy-left licenses, peer-review, OSS ideologies, tiered organization of task force, versatile online platform, and parallel code structure all contribute to the success of knowledge management.

However, past research is largely qualitative studies and focuses on a few high-profile OSS projects, such as Linux, K Desktop Environment (KDE), and Debian. Moreover, these studies confine the theoretical foundations to social and organizational learning theories (Crowston et al. 2012; Hemetsberger and Reinhardt 2006). To understand the rich phenomena of knowledge management in OSS teams, additional theoretical lenses are certainly warranted. Therefore, this study aims to gain new insights by adopting new theoretical perspectives and collecting data from a large sample of OSS teams. Specifically, this study adopts theories of transactive memory system (TMS) (Wegner et al. 1991) and compensatory adaptation (Kock 1998) to build a research model to explain knowledge sharing behaviors of OSS developers, and then tests the model against the empirical data from 120 OSS teams.

The rest of the paper is organized as follows. First, the literature on knowledge management of OSS teams is reviewed and the hypotheses are developed. Then, the research methodology is described and the results are presented. At the end, the theoretical and pragmatic implications of the findings, the limitations of the research, and future research directions are discussed.

**Literature Review**

Because of the high visibility of Linux, a number of researchers studied the Linux community in details. Ciborra and Andreu (2001) found that the community created a peculiar knowledge management model. This model facilitated knowledge sharing among the Linux developers from four aspects: first, the final product (i.e., the Linux kernel program) was a piece of codified, explicit knowledge, thus well suited to knowledge transfer. Second, the coordination among the developers was pre-specified by the software structure. Third, the Internet and associated tools help avoid communication bottlenecks. Fourth, the OSS culture promoted openness and sharing. Lee and Cole (2003) had similar findings. They reported that a community-based model of knowledge creation was operating in the Linux community. This model was characterized by the norm of open sharing, parallel code structure and two-tier task structure, and many-to-many digital communications. Such characteristics enabled the Linux community to maintain an evolutionary learning process driven by criticism and error corrections. Lanzara and Morner (2004) found that electronic communication artifacts, such as mailing lists, provided a virtual work environment for the Linux developers, where knowledge was exchanged, discussed, and evolved. Dafermos (2005) explained that the parallel code structure rendered the Linux community a parallel learning, which ensured exploration and exploitation without trading off each other. Huysman and Lin (2005) found that knowledge sharing among the Linux developers manifested itself mostly in the problem solving process. In this process, knowledge was shared not through a straightforward ask-and-answer one-way action but a continuous negotiation.

Besides Linux, researchers also examined knowledge activities in other OSS projects. Hemetsberger and Reinhardt (2006) analyzed the learning and knowledge-building processes in the KDE project from the social-experiential view. They reported that the online platform created a field enabling reflective observation, abstract conceptualization, active experimentation, and concrete experience. These all contributed to learning and knowledge creation taking place in the project. Kuk (2006) found that moderate strategic interaction among the KDE developers had a positive impact on knowledge sharing; however, extreme interaction concentration would lead to underutilizing knowledge resources of the development team. Haefliger et al. (2008) conceptualized software code reuse as one form of knowledge reuse, and reported that the developers reused code in order to set up a workable code base quickly, focus their efforts on preferred rather than mundane coding tasks, and overcome resource constraints, such as time and skills.
Although previous studies have provided important insights into knowledge management practices existing in OSS teams, there are limitations. First, all, except Kuk (2006), adopt the case-study approach. The cases examined, such as Linux and KDE, are quite unique and not representative. Thus, the findings from these studies suffer severely in terms of generalizability. Second, Crowston et al. (2012) pointed out that empirical OSS research has so far confined their theoretical foundations to social and organizational learning theories. Different theoretical lenses are strongly encouraged to gain additional understanding of the dynamics of knowledge management in OSS teams. Therefore, this study builds a positivist model to explain knowledge sharing behaviors of OSS developers and tests the model with the data from 120 OSS teams. This study chooses to focus on knowledge sharing because software development essentially is a process in which developers purposely share and integrate their individually-possessed specialized knowledge to design a software solution for a business problem (Tiwana 2004). This study chooses to employ theories of TMS and compensatory adaption as the theoretical foundation. These theories give explanations on how individuals overcome the negative effects of geographic distance and lean media on communication and coordination, and thus may provide useful theoretical perspectives to understand knowledge sharing in highly distributed OSS project teams (Espinosa et al. 2007; Kock 2004).

**Theoretical Background and Hypotheses Development**

**Knowledge Sharing among OSS Developers**

The conceptualization of knowledge sharing is tightly related to the epistemological perspectives of knowledge (Boer et al. 2002). According to Polanyi (1966), knowledge always contains two interdependent dimensions, one explicit and the other tacit. The explicit dimension refers to the part of knowledge that can be faithfully codified in words and numbers. In contrast, the tacit dimension has a personal quality, roots deeply in actions, and embeds in a specific context. Thus, tacit knowledge cannot be clearly articulated and captured, but discernible in practices of and interactions between capable practitioners (Tsoukas 2003). Polanyi’s explicit/tacit dimensions closely resemble the know-kthat/know-khow distinction of knowledge proposed by Ryle (1949). Ryle argued that know-kthat (i.e., explicit knowledge) by itself is useless. It must combines with appropriate know-khow (i.e., tacit knowledge) in order to be actionable.

Because of its two-dimensional nature, knowledge may be shared in two different modes. Explicit knowledge, on one hand, may be codified and stored in written texts, images, and language, and then be transferred as knowledge-objects in a sender-receiver model (Boer et al. 2002). Tacit knowledge, on the other hand, is personal and contextual (Polanyi 1966). Thus, it can only be shared through intense person-to-person interactions and active experimentation in the relevant context (Lam 2000). Knowledge sharing of this mode, in essence, is a social process through which diverse knowledge from different individuals is integrated to establish a shared understanding about reality (Boer et al. 2002). New knowledge often emerges in this mode of sharing: participants collectively arrive at new insights by going back and forth on issues in discussion (Hansen et al. 1999).

The two modes of knowledge sharing are discussed as if they were independent from each other. In practice, they are closely intertwined and cannot be reduced to one another. Nevertheless, depending on the particular context, one mode might be pursued predominately over the other (Hansen et al. 1999). In OSS teams, tacit knowledge sharing is pursued predominately over explicit one. This is because a large part of OSS development tasks involves resolving project issues, such as bugs, patches, and feature requests (Au et al. 2009; Mockus et al. 2002). These problem-solving activities have more to do with various know-khow and problem-solving experience resided in individual developers than standardized, codified know-kthat (Lam 2000). As Huysman and Lin (2005) pointed out, resolving a project issue is not a simple transfer of knowledge-objects between OSS developers. Rather, it involves a continuous process of integrating diverse perspectives, insights, and interpretations from different developers, combined with trial-and-error and experimentation.

**Team Knowledge–TMS**

Team knowledge is a higher level knowledge structure, where meta-knowledge of a team, such as knowledge of teamwork, taskwork, and team situation, is stored (Wildman et al. 2012). The TMS is one type of team knowledge under the category of teamwork knowledge and defined as a shared cognitive
system that team members develop to encode, store, and retrieve knowledge of different substantive domains (Nevo and Wand 2005; Ren and Argote 2011; Sharma and Yetton 2007). The TMS manifests itself through three behavioral aspects, namely, knowledge differentiation, location, and credibility (Austin 2003; Lewis 2003; Lewis 2004; Palazzolo et al. 2006). Knowledge differentiation refers to the extent to which each team member develops a distinct and non-redundant knowledge specialty rather than reproducing knowledge other members already possess. Knowledge location refers to the extent to which team members are familiar with the distribution of task-relevant knowledge in a team. Knowledge credibility refers to the extent to which team members have mutual confidence in each others’ knowledge.

One unique characteristic of an OSS team is that its developers are geographically dispersed. As a result, there are no opportunities for face-to-face interactions, a low level of presence awareness, and a high level of unfamiliarity among team members (Espinosa et al. 2007). These resultants obstruct not only the simple ask-and-answer mode of knowledge sharing (i.e., explicit knowledge sharing) but also the continuous negotiation mode of knowledge sharing (i.e., tacit knowledge sharing). However, team cognition literature suggests that geographic distance may be a double-edge sword. It may help a team develop TMS that can neutralize its negative effects on knowledge sharing (Maynard et al. 2012). Because of geographic distance, OSS developers have to rely on technological tools, such as mailing lists and CVS, to coordinate their work. Maynard et al. (2012) argued that the use of technological tools might allow for a set of better team preparation activities, including identifying a team’s main task, prioritizing goals, and developing contingency plans. These activities provide team members with a blueprint for what is to be done, how it is to be done, and who is going to do it and thus facilitate the development of TMS.

Previous empirical studies have consistently shown that the TMS leads to effective knowledge sharing among members of various teams. For example, Oshri et al. (2008) reported that the TMS promoted knowledge sharing in globally distributed software development teams. Huang (2009) reported that the TMS was positively related to knowledge sharing in R&D teams. Choi et al. (2010) reported that the TMS improved knowledge sharing in on-going organizational teams. Accordingly, this study suggests that the TMS should have a positive impact on knowledge sharing among OSS developers. This proposition provides a new theoretical explanation on how OSS developers overcome the difficulty of knowledge sharing they particularly face. The positive association between TMS and knowledge sharing is due to following reasons: 1) knowledge differentiation can help developers reasonably anticipate one another’s knowledge needs. This, in turn, prepares them for sharing their own knowledge with others in an effective manner; 2) knowledge location can reduce the duration of knowledge searching and thus enable developers to have timely and accommodated access to knowledge needed; 3) if developers do not have sufficient confidence in one another’s knowledge, they will be hesitant and unlikely to ask for help from others when encountering a task difficulty for fear that others may give incorrect suggestions. In contrast, if knowledge credibility is high, developers will more actively and openly seek suggestions and exchange ideas.

Hypothesis 1: The TMS is positively associated with knowledge sharing among OSS developers.

**The Mediating Role of Communication Quality**

In the following, it will be argued that communication quality may be one of important mediators for the relationship between TMS and knowledge sharing among OSS developers. This study argues that communication quality is positively associated with knowledge sharing among OSS developers. Besides geographical dispersion of its members, the OSS team has another unique characteristic: almost all the communication among its members is computer-mediated and text-based. Espinosa et al. (2007) suggested that face-to-face communication (FTFC) could allow team members to have frequent, rich and spontaneous communication. It is through such communication that knowledge, especially tacit
knowledge, can be successfully shared (Harada 2003). The text-based computer-mediated communication (CMC), however, suffers from the loss in richness, social presence, and spontaneity and thus introduces particular challenges for knowledge sharing.

Nevertheless, the compensatory adaptation theory (Kock 2004) suggests that OSS developers may compensate the loss in text-based CMC by improving elaborateness, thoughtfulness, and usefulness of communication. As Kock (2004) argued, human beings’ communication apparatus has naturally evolved to being suitable for FTFC. Accordingly, the text-based CMC presents a less than natural medium for our communication apparatus. The decrease in naturalness forces individuals to put more cognitive effort in communication activities and induces them to adapt their communication behaviors in a compensatory way. For instance, the message content of the text-based CMC tends to be more elaborated and well-thought due to text-based and asynchronous nature of communication. In addition, the text-based CMC, by increasing the cognitive and physical cost of communication, reduces gossiping to the minimum level and makes communication more task-oriented and objective.

Communication serves as a central means for knowledge sharing (Michailova and Sidorova 2011). Both explicit and tacit knowledge is primarily shared through communication: communication for sharing explicit knowledge involves straightforwardly transferring knowledge-objects from a sender to a receiver, whereas communication for sharing tacit knowledge demands devious and intense person-to-person interactions (Boer et al. 2002). Hence, the loss of richness, social presence, and spontaneity in the text-based CMC definitely produces detrimental impacts on knowledge sharing among OSS developers. However, as the compensatory adaptation theory suggests, the developers may improve elaborateness, thoughtfulness, and usefulness of communication to counteract such detrimental impacts. Therefore, this study argues that elaborateness, thoughtfulness, and usefulness of communication among OSS developers are positively associated with knowledge sharing among them. In addition, even though asynchronization of CMC allows the developers more time for thinking thoroughly and carefully about what others have said and what they are going to communicate, asynchronization must be confined to limited time range. Otherwise, it may cause harm because long delay between communication participants pauses knowledge flow. Therefore, this study also argues that the timeliness of communication among the developers is positively associated with knowledge sharing among them.

Hypothesis 2a: The quality of communication among OSS developers is positively associated with knowledge sharing among them.

TMS and Communication Quality

This study argues that the TMS can help OSS developers improve communication quality. Research on communication in teams reports that members tend to focus on and communicate knowledge they have in common rather than knowledge each member uniquely holds (Lee et al. 2014). This type of communication bias certainly reduces the usefulness of communication because members do not gain any new knowledge through such communication. Knowledge differentiation and location can effectively mitigate this bias since who-uniquely-knows-what informs members with whom they should communicate to gain knowledge they do not have. Thus, knowledge differentiation and location can enhance the usefulness of communication. Knowledge location may also promote timely communication. For example, suppose there is a team with members A, B, C, D, E, F, and G. E has knowledge that A needs. If knowledge location is not known, E maximally needs to inquiry six times in order to reach A. However, if knowledge location is known, E only needs to inquire once. Thus, the timeliness of communication is improved. Lastly, knowledge credibility can reduce the overheads of communication. A low level of knowledge credibility among team members leads to two consequences: first, when a member communicates knowledge to another member, he will need to communicate not only knowledge itself but also explain why knowledge provided is accurate and applicable; second, after a member receives knowledge from another member, he may choose to communicate with a third member in order to verify the accuracy of knowledge received. These overheads can be eliminated when knowledge credibility is high. This, in turn, raises the timeliness and usefulness of communication. Applying the above discussion to the OSS team context, this study hypothesizes:

Hypothesis 2b: The TMS is positively associated with the quality of communication among OSS developers.
TMS, Communication Quality, and Knowledge Sharing

In the development of Hypothesis 1, 2a, and 2b of this study, TMS is predicted to be related to knowledge sharing and communication quality, and communication quality is predicted to be related to knowledge sharing. This study, therefore, proposes that communication quality should partially mediate the relationship between TMS and knowledge sharing. This study proposes a partial rather than full mediation effect because the text-based CMC is an important means for knowledge sharing among OSS developers but not the only means. For example, one developer submits a new piece of code into the code repository. Other developers can learn the coding techniques embedded this code by studying it. This is another way to share knowledge.

Hypothesis 2c: Communication quality partially mediates the relationship between TMS and knowledge sharing among OSS developers.

Research Method

Sample and Data Collection

The sample of OSS project teams was drawn from Sourceforge (www.sourceforge.net), one of the largest OSS online communities in the world. The projects were broadly classified into fourteen categories: Clustering, Database, Desktop, Development, Enterprise, Financial, Games, Hardware, Multimedia, Networking, Security, System Administration, Storage, and VoIP. Given the practical difficulty of sampling projects from all fourteen categories, two categories (i.e., Networking and Development) were randomly selected as the sample pool. The projects in the selected categories must have met two criteria in order to be included in the study. First, the project sampled must have at least two developers since this study examined knowledge sharing between members of an OSS team. There were many projects with a single developer, and they were deleted from the sample pool. Second, to make sure that abandoned projects were excluded, the project must have been active (e.g., having new commit logs in the code repository) at the time of sampling. Two-hundred-sixteen (216) projects met the criteria. Ninety-three (93) projects were from the Networking category, and one-hundred-twenty-three (123) were from the Development category. A series of three emails were sent out, enclosing a Web survey link, to invite 703 developers of chosen projects to participate in the study. One hundred-fifty-five (155) developers completed the survey, yielding a response rate of 22.05%. They were from 120 projects. Because of IRB's restrictions on personal information, demographic information from these respondents was not collected.

Variables and Measures

Table 1 lists the measurement items used in this study. They were all adapted from previous studies. Specifically, the items from Lewis (2003) and Kanawattanachai and Yoo (2007) were adapted to measure three dimensions of TMS. The original items were developed in the context different from the current study. Therefore, the items were reworded. Communication quality was measured using the items from Stewart and Gosain (2006). The original items were developed in the context of OSS project teams, so they fit the current study well. Knowledge sharing was measured using the items from Faraj and Sproull (2000). The original items were developed to measure knowledge sharing between members of software development teams in a high-tech firm. Therefore, these items were also reworded to fit the current study context. Respondents were asked to rate the items based on their interactions with other members, using a 7-point Likert scale anchored from “strongly disagree” to “strongly agree.”
Table 1. Survey Items

<table>
<thead>
<tr>
<th>Construct</th>
<th>Scale Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge Differentiation (KD)</td>
<td>KD1: Each team member has specialized knowledge of some aspect of our project.</td>
</tr>
<tr>
<td></td>
<td>KD2: Different team members are responsible for different domains of expertise needed for our project.</td>
</tr>
<tr>
<td>Knowledge Location (KL)</td>
<td>KL1: Our team has a good “map” of each member’s talents and skills.</td>
</tr>
<tr>
<td></td>
<td>KL2: Members on our team know what task-related skills and knowledge they each possess.</td>
</tr>
<tr>
<td></td>
<td>KL3: Members on our team know who has specialized skills and knowledge that is relevant to their work.</td>
</tr>
<tr>
<td>Knowledge credibility (KCR)</td>
<td>KCR1: The members on our team trust that the other members’ knowledge about the project is credible.</td>
</tr>
<tr>
<td></td>
<td>KCR2: The members on our team are confident when applying the knowledge provided by other members to the project tasks at hand.</td>
</tr>
<tr>
<td></td>
<td>KCR3: The members on our team did not have much faith in the other members’ expertise (reversed).</td>
</tr>
<tr>
<td>Communication quality (CQ)</td>
<td>CQ1: Members on our team answer each other’s questions in a timely manner.</td>
</tr>
<tr>
<td></td>
<td>CQ2: Our team members’ responses to each other’s questions are correct and useful.</td>
</tr>
<tr>
<td></td>
<td>CQ3: Members on our team answer each other’s questions in a thoughtful manner.</td>
</tr>
<tr>
<td>Knowledge sharing (KS)</td>
<td>KS1: Members in our team share their special knowledge and expertise with one another.</td>
</tr>
<tr>
<td></td>
<td>KS2: Members in our team virtually do not share their information, knowledge, or skills with one another (reversed).</td>
</tr>
<tr>
<td></td>
<td>KS3: More knowledgeable members in our team willingly make their knowledge and expertise available to other members.</td>
</tr>
</tbody>
</table>

Analytical Techniques

This study employed partial least square (PLS) for data analysis. In contrast to the covariance-based structural equation modeling (SEM) (e.g., AMOS and LISREL), the component-based PLS estimates model parameters by minimizing the residual variances of dependent variables (Lohmöller 1984; Lohmöller 1989). PLS does not have the issues of inadmissible solutions and factor indeterminacy, from which the covariance-based SEM often suffers (Chin 1998). In addition, it makes minimal assumptions on measurement scales, sample size, and residual distributions (Chin 1998; Wold 1985). Thus, PLS was chosen as the statistical method of data analysis.
Analysis and Results

Measurement Validation

Because the TMS is a multi-dimensional construct, this study constructed the corresponding PLS model following the method recommended by Wetzels et al., (2009). The model is shown in Figure 1. The discriminant validity was first assessed by loadings and cross-loadings of the survey items. Table 2 shows that the items load highly (i.e., greater than 0.80) on their intended constructs, and the gaps between the cross-loadings are greater than 0.1 (Nunnally, 1978). The evidence of discriminant validity was also obtained by evaluating the square root of average variance extracted (AVE). As shown in Table 3, the square root of AVE for each latent variable (i.e., values on the diagonal) is greater than the correlations between latent variables (Chin 1998). The reliability was tested in terms of composite reliability (CR) and Cronbach’s Alpha, which are all above the recommended levels (shown in Table 3).

![Figure 1. The Second-order Factor Model](image)

Table 2. Loadings and Cross-loading of the Survey Items

<table>
<thead>
<tr>
<th></th>
<th>KD</th>
<th>KL</th>
<th>KCR</th>
<th>CQ</th>
<th>KS</th>
</tr>
</thead>
<tbody>
<tr>
<td>KD1</td>
<td>0.91</td>
<td>0.44</td>
<td>0.39</td>
<td>0.31</td>
<td>0.19</td>
</tr>
<tr>
<td>KD2</td>
<td>0.89</td>
<td>0.43</td>
<td>0.26</td>
<td>0.31</td>
<td>0.14</td>
</tr>
<tr>
<td>KL1</td>
<td>0.35</td>
<td>0.85</td>
<td>0.40</td>
<td>0.42</td>
<td>0.33</td>
</tr>
<tr>
<td>KL2</td>
<td>0.40</td>
<td>0.89</td>
<td>0.42</td>
<td>0.38</td>
<td>0.32</td>
</tr>
<tr>
<td>KL3</td>
<td>0.51</td>
<td>0.90</td>
<td>0.43</td>
<td>0.46</td>
<td>0.32</td>
</tr>
<tr>
<td>KCR1</td>
<td>0.31</td>
<td>0.41</td>
<td>0.92</td>
<td>0.44</td>
<td>0.37</td>
</tr>
<tr>
<td>KCR2</td>
<td>0.27</td>
<td>0.35</td>
<td>0.90</td>
<td>0.44</td>
<td>0.37</td>
</tr>
<tr>
<td>KCR3R</td>
<td>0.38</td>
<td>0.49</td>
<td>0.84</td>
<td>0.40</td>
<td>0.28</td>
</tr>
<tr>
<td>CQ1</td>
<td>0.28</td>
<td>0.42</td>
<td>0.42</td>
<td>0.85</td>
<td>0.33</td>
</tr>
<tr>
<td>CQ2</td>
<td>0.35</td>
<td>0.41</td>
<td>0.46</td>
<td>0.88</td>
<td>0.28</td>
</tr>
<tr>
<td>CQ3</td>
<td>0.28</td>
<td>0.42</td>
<td>0.38</td>
<td>0.88</td>
<td>0.41</td>
</tr>
<tr>
<td>KS1</td>
<td>0.17</td>
<td>0.36</td>
<td>0.34</td>
<td>0.39</td>
<td>0.94</td>
</tr>
<tr>
<td>KS2R</td>
<td>0.20</td>
<td>0.35</td>
<td>0.36</td>
<td>0.40</td>
<td>0.93</td>
</tr>
<tr>
<td>KS3</td>
<td>0.11</td>
<td>0.27</td>
<td>0.32</td>
<td>0.24</td>
<td>0.82</td>
</tr>
</tbody>
</table>
Table 3. Descriptive Statistics, Reliabilities, and Correlations

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>S.D.</th>
<th>CR</th>
<th>Alpha</th>
<th>KD</th>
<th>KL</th>
<th>KCR</th>
<th>CQ</th>
<th>KS</th>
<th>TMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>KD</td>
<td>5.17</td>
<td>1.26</td>
<td>0.89</td>
<td>0.77</td>
<td>0.90*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KL</td>
<td>5.22</td>
<td>1.09</td>
<td>0.91</td>
<td>0.86</td>
<td>0.48</td>
<td>0.88*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KCR</td>
<td>5.93</td>
<td>0.83</td>
<td>0.92</td>
<td>0.87</td>
<td>0.36</td>
<td>0.47</td>
<td>0.89*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CQ</td>
<td>5.66</td>
<td>1.11</td>
<td>0.91</td>
<td>0.84</td>
<td>0.35</td>
<td>0.48</td>
<td>0.48</td>
<td>0.87*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KS</td>
<td>5.96</td>
<td>1.11</td>
<td>0.93</td>
<td>0.88</td>
<td>0.18</td>
<td>0.37</td>
<td>0.38</td>
<td>0.39</td>
<td>0.90*</td>
<td></td>
</tr>
<tr>
<td>TMS</td>
<td>5.54</td>
<td>0.80</td>
<td>0.89</td>
<td>0.86</td>
<td>0.69</td>
<td>0.85</td>
<td>0.81</td>
<td>0.56</td>
<td>0.41</td>
<td>0.71*</td>
</tr>
</tbody>
</table>

*Values on the diagonal are the square-root of average variance extracted for each construct.

Common Method Variance

The data of this study was collected from a single set of respondents. Hence, common method factor (Podsakoff et al. 2003) was used to test whether the common method variance (CMV) was a problem. The results (because of the page limitation, the test results are not included here) revealed that the substantive factor loadings were far greater than the method factor loadings and the method factor contributed very little to variances of the indicators. Therefore, the CMV was not a serious concern.

Hypothesis Testing

The hypotheses were tested by examining path coefficients of the PLS structural model. Hypothesis 1 proposes a positive relationship between TMS and knowledge sharing among OSS developers. The results are revealed in Figure 2. The corresponding path coefficient is positive and statistically significant ($\beta=0.28$, $t=2.78$, $p<0.01$), thus supporting Hypothesis 1. A global fit measure for the model shown in Figure 2 was calculated. This global fit measure is defined as the geometric mean of the average communality and average $R^2$ for endogenous constructs (Tenenhaus et al. 2005). The value obtained is 0.43, greater than the large effect size cut-off value of 0.36 (Wetzels et al. 2009), indicating a good fit of the model.

Hypothesis 2a proposes a positive relationship between communication quality and knowledge sharing. The corresponding path coefficient, as shown in Figure 2, is positive and statistically significant ($\beta=0.23$, $t=2.08$, $p<0.05$), thus supporting Hypothesis 2a. Hypothesis 2b proposes that the TMS should have a positive impact on communication quality. The corresponding path coefficient, as shown in Figure 2, is positive and statistically significant ($\beta=0.56$, $t=5.86$, $p<0.01$). Therefore, Hypothesis 2b is supported. Hypothesis 2c proposes that the effect of TMS on knowledge sharing should be partially mediated through communication quality. To test this hypothesis, the Baron-Kenny steps were followed (1986). First, the model without communication quality (shown in Figure 3) was examined. The results reveal that the relationship between TMS and knowledge sharing is positive and statistically significant ($\beta=0.41$, $t=3.87$, $p<0.01$). A global fit measure for the model shown in Figure 3 was also calculated. The value obtained is 0.33, greater than the medium effect size cut-off value of 0.25 (Wetzels et al. 2009), indicating a good fit of the model. Second, the same relationship was examined after communication quality was included in the model (shown in Figure 3). The results show that the path coefficient of the relationship reduces to 0.28 ($t=2.78$, $p<0.01$) from 0.41. The t-test indicates that the change is statistically significant ($t(308)=10.40$, $p<0.01$). Based on the above results, it is concluded that Hypothesis 2c is supported.
Discussion

The purpose of the current study is to gain new insights on knowledge sharing in the context of OSS teams. A research model, based on team cognition literature and compensatory adaptation theory, is proposed and confirmed using data from 120 OSS teams. First, this study finds that the TMS is positively associated with knowledge sharing among OSS developers. The positive association is expected because knowledge differentiation prepares the developers to effectively respond to each others’ inquiries;
knowledge location helps lower the cost of knowledge searching; knowledge credibility enables the developers to actively and openly exchange ideas (Ren and Argote 2011).

Second, this study finds that communication quality is positively associated with knowledge sharing among OSS developers. This association can be attributed to compensatory adaptation behaviors commonly observed in the text-based CMC. People share knowledge as they communicate with each other in a social context (Lam 2000). Even though the human communication apparatus has naturally evolved to fit FTFC (Kock 2004), the human brain is fairly malleable such that it is able to develop new schemas by modifying the neocortex (Pinker 1997; Wills 1993). Therefore, individuals can learn to adapt to the text-based CMC by improving communication quality (Kock 2004). The improvement of communication quality compensates the lack of social presence and communication richness in the text-based CMC as the content of communication is more focused, well thought-out, and clearly articulated. The focused, well thought-out, and clearly articulated communication certainly helps knowledge sharing among the developers.

Third, this study finds that the TMS is positively related to communication quality of developers. The positive relationship is due to three reasons: 1) knowledge differentiation can reduce the communication bias favoring commonly rather than uniquely held knowledge and thus enhance the usefulness of communication (Lee et al. 2014); 2) knowledge location can minimize the time spent for knowledge search and thus improve the timeliness of communication (Hollingshead and Brandon 2003); 3) knowledge credibility can reduce the overheads of communication, such as justifying courses of actions and criticizing each other’s work (Moreland 1999), and thus again enhance the usefulness of communication.

Lastly, this study finds that communication quality partially mediates the relationship between TMS and knowledge sharing. This is because the text-based CMC is one of important means for sharing knowledge among OSS developers. As Hemetsberger and Reinhardt (2006) described, after an initial idea for software development is presented in the mailing list, lively text-based CMC ensues. Some developers support and further elaborate on the idea. Others point out the flaws and provide improvement. The idea is bounced back and forth, resulting in a snowball effect. At the end, the initial idea evolves into something none of the developers thought of at the beginning.

Contributions to Literature

Previous studies reported that explicit mechanisms, such as software structure and collaboration tools, were effective for overcoming knowledge management difficulties in OSS teams (Ciborra and Andreu 2001; Lanzara and Morner 2004). This study, on the other hand, examines two implicit mechanisms—TMS and communication quality. The finding of their positive impacts on knowledge sharing suggests that future research should pay more attention to such implicit mechanisms. Moreover, this study finds that such implicit mechanisms do not function separately. Rather, they interactively affect knowledge sharing. For instance, the TMS promotes communication quality, which, in turn, improves knowledge sharing.

While the importance of communication quality has been established in the OSS context, there is a lack of research on the means by which quality communication is achieved (Stewart and Gosain 2006). The finding of the positive association between TMS and communication quality suggests that once the TMS is developed, OSS developers may minimize communication volume and frequency because they can communicate smartly (Kanawattanachai and Yoo 2007). Therefore, developing the TMS is an effective means to improve communication quality.

The literature on compensatory adaptation has so far focused on explaining why such behavior takes place in the CMC context. However, dependent variables of compensatory adaptation is lacking from the theory. The finding of the positive relationship between communication quality and knowledge sharing suggests that knowledge sharing should be one of such variables.

Implications for Practice

The results of this study can offer several practical suggestions to OSS project administrators and also organizations experimenting with the open source mode of software development (e.g., Google and
Microsoft): first, knowledge sharing among the developers can be improved by helping them become familiar with each other’s expertise. This can be done by implementing a knowledge map or directory on the project website, indicating each member’s areas of expertise. Organizations, such as IBM, have used such a directory in its internal knowledge portal, serving as a way to help knowledge workers familiarize themselves with their colleagues (Mack et al. 2001). Second, given the impact of communication quality on knowledge sharing, project administrators should pay particular attention to improving communication quality between OSS developers. The asynchronous, text-based lean media, such as mailing list, can lead to compensatory adaption, which gives rise to communication quality. Therefore, project administrators may purposely prevent real-time communication tools from being implemented in order to force OSS developers to invest more cognitive efforts into communication.

**Limitations and Future Research**

Despite its contributions, this study has several limitations. First, two unique characteristics of OSS teams, geographic dispersion and text-based CMC, are discussed. This study argues that these characteristics cause TMS and communication quality to arise among OSS developers. However, this study does not test the relationships between these characteristics and TMS and communication quality. One direction for future research is to empirically examine the relationships. Second, this study argues and tests the importance of TMS and compensatory adaptation in knowledge sharing among OSS developers. Future research may extend the arguments to other team contexts and perform a comparative analysis among co-located, distributed, and OSS teams. Lastly, the sample of this study was exclusively drawn from two project categories on the Sourceforge website: Networking and Development, which limits the generalizability of the study’s findings. Thus, another direction for future research is to replicate this study in other project categories of Sourceforge.

**Acknowledgements**

The author thanks anonymous reviewers and the associated editor for their helpful comments on this article. This work was supported by a grant from the National Natural Science Foundation of China (71402144).

**References**


Knowledge Sharing in OSS Project Teams


