International Journal of Information Management 33 (2013) 553–563

Contents lists available at SciVerse ScienceDirect

International Journal of Information Management

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

Knowledge sharing in open source software project teams: A transactive memory system perspective

Xiaogang Chen[a,*], Xue Li[b], Jan Guynes Clark[c], Glenn B. Dietrich[c]

[a] Southwestern University of Finance and Economics, China
[b] Shaanxi Normal University, China
[c] The University of Texas at San Antonio, USA

ARTICLE INFO

Article history:
Available online 23 February 2013

Keywords:
Knowledge sharing
Transactive memory system
Communication quality
Technical achievement

ABSTRACT

The extant studies have not empirically examined the possible team cognitive mechanisms that facilitate knowledge sharing in OSS teams, even though knowledge sharing is a cognitive task and an OSS team is a complex cognitive system. To fill this research gap, we adopt the perspective of transactive memory system (TMS) to explore the relationships among TMS, knowledge sharing, communication quality, and technical achievement of OSS teams. By analyzing data from 95 OSS projects with the partial least squares (PLS) method, our study demonstrates that several TMS dimensions have positive impacts on knowledge sharing behaviors and communication quality. Moreover, communication quality positively influences technical achievement of OSS teams. These findings provide useful implications for literature and practice.

1. Introduction

Open source software (OSS) has generated much excitement in the software market. Companies have been considering OSS as a viable and economic substitution for proprietary software to support their business processes for quite some time (Andersen-Gott, Ghinea, & Bygstad, 2012). The New York Stock Exchange (NYSE) adopted Linux recently to support its electronic trading platform due to low cost, flexibility, and high level of security associated with Linux (Asay, 2008). Quite a few software producing firms, such as Red Hat, Geeknet, and Mozzilla, have centered their business models on OSS. Even Microsoft, the leading ideological opponent of the OSS community, launched its own OSS hosting site, CodePlex, in July 2006 (Voth, 2006).

The OSS development differs significantly from the traditional in-house software development in three aspects: first, OSS is distributed under copy-left licenses, which guarantee that everyone has free and unrestrictive access to the source code of software (Colazo, Fang, & Neufeld, 2005; Waring & Maddocks, 2005). On the other hand, the traditional software is distributed under copyright licenses. Such licenses only grant the license holders the access to the source code, and others are only able to view the binary code. Therefore, the traditional software is often called closed source software (CSS). Second, whereas CSS is developed by organizational teams with commercial purposes, OSS is largely built and maintained by teams of voluntary developers. These developers are self-motivated by altruism, learning purpose, career advancement, and personal software need (Bonaccorsi & Rossi, 2003; Lerner & Tirole, 2002; Raymond, 2001; Ye & Kishida, 2003). Third, OSS developers are geographically dispersed and rarely, if ever, meet face-to-face (Feller & Fitzgerald, 2002). They have to rely completely on Internet tools, such as mailing lists and concurrent version control systems, to communicate and collaborate with each other (Wayner, 2000). Therefore, an OSS team is viewed as a form of distributed teams with pure virtualness (Alborts, Ramos, & Hervas, 2008; Griffith, Sawyer, & Neale, 2003), or pure virtual teams in Griffith et al.'s term. Although the developers of CSS teams may be also geographically distributed, the face-to-face communication is generally available at some points of the development cycle. Thus, CSS teams have far less degree of virtualness than OSS teams and may be best classified as hybrid teams (Griffith et al., 2003).

In spite of significant differences in licensing schemes, motivations, and virtualness, one thing remains the same for both OSS and CSS teams; that is, the knowledge intensive nature of software development. 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 & Lindvall, 2002). Software development essentially is a process in which developers purposely share and integrate their individually possessed specialized knowledge to design a software solution for an applied problem (Tiwana, 2004). Therefore, effective knowledge sharing is the key for the

* Corresponding author. Tel.: +86 28565874.
E-mail address: chenxg@swufe.edu.cn (X. Chen).

0268-4012/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ijinfomgt.2013.01.008

success of software development. This is true for both OSS and CSS teams. However, OSS teams face particular knowledge sharing challenges due to the highly distributed environment and self-directed workforce (Crowston, Wei, Howison, & Wiggins, 2012). The fact that hundreds of OSS with good quality has been successfully produced (Stamelos, Angelis, Oikonomou, & Bleris, 2002) suggests that some OSS teams cope with the challenges well. Consequently, a stream of research has emerged to examine the mechanisms that help overcome knowledge sharing barriers in OSS teams. For example, Ciborra and Andreu (2001) studied the case of the Linux community and found that four particular characteristics of the community contribute to the effectiveness of knowledge sharing: first, the final product (i.e., the Linux kernel program) is a piece of codified, explicit knowledge, thus well suited to knowledge transfer; second, the coordination needed in the development process is pre-specified by the software structure; third, the Internet and associated tools help avoid communication bottlenecks; lastly, the OSS culture promotes openness and sharing. Lanzara and Morner (2004) paid particular attention to digital communications in the Linux community. They argued that electronic communication artifacts, such as mailing lists, provide a virtual work environment, where knowledge is exchanged, discussed, and evolved. From the perspective of strategic interaction, Kuk (2006) investigated knowledge sharing in the KDE project (i.e., the graphic user interface for UNIX stations). He reported that strategic interaction has a positive impact on knowledge sharing. However, extreme interaction concentration leads to underutilizing knowledge resources. Sowe, Stamelos, and Angelis (2008) analyzed conversations on the mailing list of the Debian project and revealed that knowledge sharing activities might be best explained in terms of fractal cubic distributions. Gächter, von Krogh, and Haefliger (2010) applied the game theory perspective to research why OSS developers initiate knowledge sharing. They found that economic incentives and social preference might play important roles.

Despite valuable insights into knowledge sharing behaviors of OSS developers, the extant studies are largely focused on a few successful cases, such as Linux, KDE, and Debian. It is questionable whether their findings are generalizable to the large population of OSS projects. More importantly, knowledge sharing is a cognitive task and an OSS team is a complex cognitive system (Sowe, Stamelos, & Angelis, 2006), but existing studies have not empirically examined the possible team cognitive mechanisms that facilitate knowledge sharing in OSS teams. Team cognition literature suggests that transactive memory system (TMS), as a team cognitive mechanism, is especially useful in highly distributed teams because it synchronizes members' communication "based on unspoken assumptions about what others in the group are likely to do" (Wittenbaum & Stasser, 1996, p. 3). TMS refers to a team-level knowledge-holding structure, where various knowledge possessed by individual team members is stored and connected through the shared awareness about who-knows-what within the team (Wegner, 1987). Previous studies reported that TMS helps: (1) allocate knowledge responsibilities among the members, so that the group's cognitive resource is better utilized (Hollingshead, 1998a, 1998b); (2) identify the location of knowledge within the group, thus the members have timely and accommodated access to the knowledge needed (Lewis, 2004); and (3) reasonably anticipate one another's knowledge needs and plan knowledge sharing behaviors accordingly (Moreland, 1999; Moreland, Argote, & Krishnan, 1996). Hence, TMS is a useful team cognitive mechanism that may explain the effectiveness of knowledge sharing in distributed team in general and OSS teams in specific. Several researchers suggested the working of such mechanism in their case studies of OSS projects (e.g., Gutwin, Penner, & Schnerder, 2004; Hemetsberger & Reinhardt, 2006). Therefore, we aim to fill the research gap by applying the perspective of TMS to explore knowledge sharing behaviors in a large number of OSS teams.

The rest of the paper is organized as follows. We will first review the theory of TMS and develop the hypotheses. We will then describe the research methodology and present the results. At the end, we will discuss the theoretical and pragmatic implications of the findings, the limitations of the research, and suggest future research directions.

2. Theoretical background

Wegner (1987; see also Wegner, Giuliano, & Hertel, 1985) proposed the concept of TMS to describe the cognitive interdependence in a group of people having close relationships, such as dating couples, families, and workgroups. In such relationships, the group members tend to divide cognitive tasks on the basis of knowledge responsibility. A member incurs the responsibility for a certain knowledge domain if he or she is (1) perceived as the group's expert of the domain, (2) known to have access to the domain, or (3) assigned by an authority (e.g., the group leader). Such responsibility means that the group will channel any new information pertaining to the given knowledge domain to the member. The group will also consult the member when any questions related to the domain arise. As a result, this member becomes the source and repository of this knowledge domain for the group. Likewise, other members might incur responsibilities of other knowledge domains, and therefore specialize in those domains. Eventually, a group-level knowledge-holding system emerges. The system not only stores specialized knowledge from individual members but also the location information of knowledge. Wegner termed this system TMS and defined it as a set of individual memory systems in combination with the shared awareness about information location among the group members.

Early empirical research on TMS focused on co-located teams. For example, Liang, Moreland, and Argote (1995) found that the groups with the members trained together score higher on TMS than the groups with the members trained individually. Moreover, the higher score on TMS is responsible for the better group performance. Hollingshead (1998a) revealed that the members of co-located teams, by communicating information about each other's domains of expertise, are able to form the TMS. More recent studies begin to investigate the development of TMS in virtual teams. Kanawattanachai and Yoo (2007) reported that the members of virtual teams develop the TMS through computer-mediated communication. Oshri, van Fenema, and Kotlarsky (2008) reported that frequent teleconferencing secessions also support the TMS development in virtual teams. Choi, Lee, and Yoo (2010) suggested that information technologies, such as search engine, document repositories, and online discussion boards, are conductive to dialogic practices, which, in turn, help the TMS development in virtual teams. In the context of OSS teams, several researchers suggested the working of TMS. For example, Gutwin et al. (2004) conducted a case study of three OSS projects (NetBSD, Apache httpd, and Subversion) and found that the developers of these projects, through reading mailing-list postings and participating in on-line discussion, are able to form and maintain a general perception about the knowledge distribution within the teams. Hemetsberger and Reinhardt (2006) studied knowledge learning and building activities in the KDE project. They found that the developer mailing list serves as the proxy of TMS to help the developers locate knowledge needed. Based on the literature discussed above, we argue that it is appropriate to apply the theory of TMS to the context of OSS teams.

Prior research has consistently demonstrated that the TMS is a multi-dimensional construct. Three dimensions are most often discussed in the literature: knowledge differentiation,