2007

Antecedents of Coordination Effectiveness of Software Developer Dyads from Interacting Teams: An Empirical Investigation

Minghui Yuan
City University of Hong Kong, y31133@hotmail.com

Doug Vogel
City University of Hong Kong, isdoug@cityu.edu.hk

Xi Zhang
USTC-CityU Joint Research Institute, xizhang@mail.ustc.edu.cn

Zhenjiao Chen
USTC-CityU Joint Research Institute, Sharon@mail.ustc.edu.cn

Xuelin Chu
University of Science and Technology of China, xlchu@ustc.edu.cn

78. Antecedents of Coordination Effectiveness of Software Developer Dyads from Interacting Teams: An Empirical Investigation

Minghui Yuan
Department of Information Systems
City University of Hong Kong
y31133@hotmail.com

Doug Vogel
Department of Information Systems
City University of Hong Kong
isdoug@cityu.edu.hk

Xi Zhang
USTC-CityU Joint Research Institute
Suzhou, P.R.China
xizhang@mail.ustc.edu.cn

Zhenjiao Chen
USTC-CityU Joint Research Institute
Suzhou, P.R.China
Sharon@mail.ustc.edu.cn

Xuelin Chu
USTC Suzhou Institute for Advanced Study
University of Science and Technology of China
xlchu@ustc.edu.cn

Abstract
Among numerous reasons for software project failure, coordination failure is considered as especially salient. Prior studies on coordination in software development are confined to team internal coordination and do not explicitly differentiate team internal and external coordination processes. This study proposes a research model to explain the antecedents of coordination effectiveness of software developer dyads from interacting teams. We explore the antecedents by integrating inter-personal coordination and technology-based coordination. Data were collected from 59 software developer dyads from different interacting teams as well as software developers' managers. The results reveal that implicit knowledge sharing has a significant positive impact on coordination effectiveness. Social capital (mutual trust and project commitment) has a significant impact on knowledge sharing with mutual trust directly affecting both implicit and explicit knowledge sharing. Project commitment also has a direct impact on explicit knowledge sharing and mutual trust, but it does not directly affect implicit knowledge sharing.

Keywords: Software development, Team-external coordination, Software developer dyads, Coordination theory, Social capital theory

Introduction

Coordination in software development is often not confined to team-internal coordination. Intergroup coordination is inevitably involved in some larger software projects. Due to the complexity of software projects, they exceed the capacity of any individual or one single team. In such situation, software projects are divided into different subsystems. Every subsystem may include one or several modules. Thus, multiple software development teams are involved. Due to the thin spread of application domain knowledge (Curtis et al. 1988), it is very difficult, if not impossible, for developers to understand the modules that are developed by other teams. So software developers in one team are often required to work closely with other developers in other teams when system-level requirements or issues occur.

Coordination is also extended from team-internal coordination to intergroup coordination. However, extra costs may occur in intergroup coordination. As a rough rule, three groups of three programmers can do only twice the work of a single group (or four times the work of a single programmer) because of the time for coordination (Weinberg 1998). Further, bad intergroup coordination often leads to very serious consequences, such as incompatible schedules, system-level design and requirements defects, and system-level problems, etc. So it is very important to effectively manage the intergroup coordination. However, it is by no means an easy task.

Prior coordination studies in software development are confined to team internal coordination. Even some studies involve projects with multiple teams; they do not explicitly differentiate team internal and external coordination processes. The relationship between team-internal and team-external cooperative processes remains quite ambiguous (Ancona 1990; Hoegl et al. 2004).However, there should be some differences between team-internal and team-external coordination. First, In terms of social identity theory (Tajfel 1981; Tajfel et al. 1986; Turner 1984), there is a lot of evidence that "ingroup favoritism" exists: Usually, people view ingroup members more positively than outgroup members, and evaluate ingroup members as more trustworthy, honest, loyal, cooperative, and valuable to the group than outgroup members (Brown 2000; Hewstone et al. 2002; Kane et al. 2005). In addition, functional diversity may also lead to different outlooks, perceptions, and work procedures between two or more software development groups working in the same project. These differences may constitute barriers that hinder cross-unit contact and work coordination (Fenema 2002; Tushman et al. 1978). From this point of view, team-external coordination may be more difficult than team-internal coordination. Since numerous studies focus on team-internal coordination, we need better understandings of team-external coordination process.

This study aims at furthering our understanding of coordination of software developers from interacting teams in collocated projects where developers are located in a single physical place. The overall research question is to investigate the antecedents of coordination effectiveness of software developer dyads from interacting teams. It is different from intergroup coordination in CMM where it is defined as involving a software engineering group's participation with other project engineering groups to address system-level requirements, objectives, and issues (Paulk et al. 1993). It is at group level. While in our study, it is at individual dyadic level. We believe it is the most basic unit of cross-team coordination. Before we investigate group level coordination, it is necessary to get some insights into the basic unit. This study adopts a quantitative approach. The proposed model is grounded in social capital theory (SCT) and coordination theory. Data from interacting programmer pairs and their managers are collected from a cross-sectional survey. The rest of paper is organized as follows: First, we review the existing literature and develop our model, including hypotheses. Next, we describe the research methodology (cross-sectional survey) and analysis. Finally, we present the results and discuss their implications and limitations.

Literature

Coordination Research in Software Development

In software development, coordination means that different people working on a common project agree to a common definition of what they are building, share information, and mesh their activities (Kraut 1995). Most empirical studies on coordination in collocated projects focus on the relationship between coordination mechanisms and project performance. For example, Kraut et al. (Kraut et al. 1995) investigate the interaction of coordination techniques

and structural characteristics of projects (e.g. project size) and their effects on project outcomes and coordination success. Nidumolu (1995) claimed that higher levels of both vertical and horizontal coordination lead to higher levels of overall performance. However, explicit coordination mechanisms (e.g. requirement development techniques) are not enough for software requirements development (Crowston et al. 1998). More recently, based on the studies on team cognition, a few researchers begin to empirically investigate coordination from implicit coordination point of view. Implicit coordination has been referred to as the "synchronization of member actions based on unspoken assumptions about what others in the group are like to do" (Espinosa 2002; Wittenbaum et al. 1996). Results of these studies are encouraging. Crowston and Kammerer (1998) point out a well-developed collective mind is an alternative coordination mechanism. With a well-developed collective mind, a group member can spend less time checking or asking, and know which features are needed, whom he has to consult for advice on which features to pick, etc. Faraj and Sproull (2000) propose expertise coordination has a strong relationship with team performance, and that expertise coordination processes are positively related to teams performance above and beyond traditional factors (e.g. presence of expertise, and administrative coordination). In sum, most empirical studies on coordination in software development focus on the relationship between coordination mechanisms and project performance. Related research includes how to decide an appropriate coordination level and how to utilize techniques to improve coordination process. As we can see, these studies are confined to team internal coordination. Since previous studies focus on team-internal coordination, we need to address this gap.

Coordination Theory

Coordination theory, which focused on the interdisciplinary study of coordination, is proposed by Malone and Crowston (1994). In this theory, coordination is defined as the process of managing dependencies between activities. Consistent with the definition, group action is analyzed in terms of actors performing interdependent tasks. These tasks might require or create resources of various types. One of the most important contributions of coordination theory is to offer a framework for understanding different types of dependencies and managing their interaction in practical setting (Malone et al. 1994). Van Fenema (2002) summarizes an integrative framework for coordination modes: coordination by organization design, work-based coordination, inter-personal coordination, and technology-based coordination. In this study, coordination theory serves as an overarching theory, which integrates different coordination mechanisms. Because this study focuses on individual and dyadic level team-external coordination, we will not consider the impact of coordination by organization design (e.g. lateral contacts) and work-based coordination (e.g. standards and operating procedures). Specifically, we will investigate how inter-personal and technology-based coordination affect coordination outcome of programmers from interacting teams. Inter-personal coordination relies on mutual adjustment, feedback and group meetings to adjust individuals' actions. While technology-based coordination refers to the use of functions and roles of technology to shape task performance.

Social Capital Theory (SCT)

Social capital is the aggregate of resources embedded within, available through, and derived from the network of relationships possessed by an individual or organization (Inkpen et al. 2005). Based on a thorough review of theoretical research on social capital undertaken in various disciplines, Adler and Kwon (2002) propose a conceptual model of social capital. In this model, opportunity, motivation and ability are three sources of social capital. Opportunities mainly relate to an actor's network of social ties. Motivation includes shared norms, trust, etc. Ability refers to the competencies and resources at the nodes of the network.

Opportunity-motivation-ability may lead to both benefits and risks. The benefits include broader sources of information as well as information's quality, relevance, and timeliness improvement. Social capital benefit mediates motivation and social capital value. However, the ultimate value of a given form of social capital also depends on task, symbolic and complementary capabilities contingencies. Task contingencies mean the fit between the social capital benefits and the organization's objectives. In this study, we focus on a particular subset of this model. Inter-personal coordination is explored by motivation (project commitment and mutual trust) and social capital benefits (knowledge sharing). Task contingencies correspond to task interdependence, value refers to coordination outcome.

Model Developing and Hypotheses

Figure 1 graphically shows the research model and hypotheses. Based on the study of Espinosa (2002), we define coordination effectiveness as the extent to which dependencies have been effectively managed between software developer dyads from interacting teams.

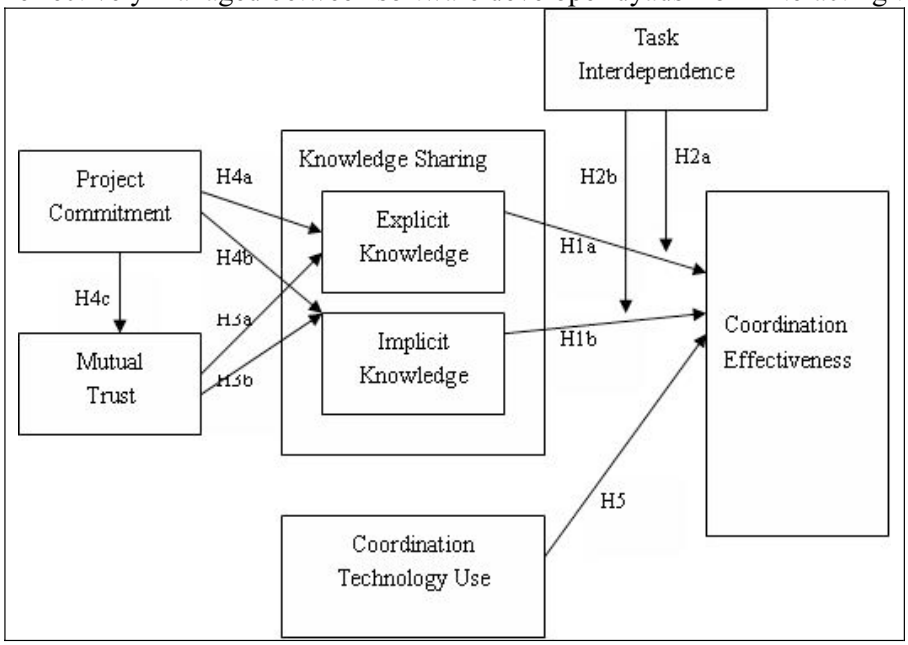

Figure 1: Research Model and Hypotheses

Knowledge Sharing (KS)

In this study, knowledge sharing is defined as activities of transferring or disseminating knowledge from one software developer to another developer in other interacting teams when engaged in a cross-team task. In software development, technical solution discussion, requirement and design review, code inspections, problem-solving, project meeting etc are all necessary to guarantee coordination effectiveness. Information processing theory asserts that increased information exchange is essential to overcome task uncertainty and task interdependence (Andres, 2002). Prior studies also suggest that knowledge sharing can effectively share domain expertise between the customer and the development team as well as capture non-externalized knowledge of the development team members, and identify the requirements of the software system (Chau 2003), which are critical to coordination effectiveness. Moreover, effective shared knowledge can be viewed as a synergy between groups (Bostrom 1989). Nelson (1996) maintains the absence of a shared reality between groups is a critical factor in dysfunctional group dynamics, while the presence of shared perception may lead to better performance. Therefore, we propose

Hypothesis 1a (H1a): Explicit knowledge sharing is positively associated with coordination

effectiveness of software developer dyads from interacting teams.
Hypothesis 1b (H1b): Implicit knowledge sharing is positively associated with coordination effectiveness of software developer dyads from interacting teams.

Task Interdependence (TI)

Based on the definition given by Andres et al. (2002), task interdependence is defined as to the extent to which a cross-team task requires software developer dyads from interacting teams to engage in workflow exchanges of information, skills, or resources, and to where actions taken by one software developer affect the actions and work outcomes of another developer. In terms of coordination theory, the interdependence includes shared resources, producer/consumer relationships, simultaneity constraints, and task/subtask dependencies (Malone et al. 1994). As teams depend on other teams' input for accomplishing their own task, the work in one team has implications for the work and progress in other teams (Hoegl et al. 2004).The most obvious example is the software interface provided by a team but used by other teams. Often the complex systems have higher task interdependence compared with less complex ones. But the design of systems also plays a critical role in reducing the interdependence. If the architecture of the systems is well designed, the loose coupling of modules will be achieved; accordingly the task interdependence will be reduced. In the higher task interdependence, greater frequency and volume of information exchange and mutual decision-making are required (Andres, 2002). If two developers have low task interdependence; they can accomplish the task with much less knowledge sharing. Therefore, we conclude
Hypothesis 2a (H2a): Task interdependence moderates the relationship between explicit knowledge sharing and coordination effectiveness of software developer dyads from interacting teams.
Hypothesis 2b (H2b): Task interdependence moderates the relationship between implicit knowledge sharing and coordination effectiveness of software developer dyads from interacting teams.

Mutual Trust (MT)

More specific to this study, mutual trust is defined as the extent to which there is reciprocal trust between the developer dyads from interacting teams. In terms of social identity theory, social categorization may lead to distrust between individuals from different groups within an organization. People tend to evaluate outgroup members as less honest, reliable, open, and trustworthy than members of their own group. Previous literature provides considerable evidence that trust plays a positive role in knowledge sharing. For example, Nelson et al. (1996) suggest that trust has a major impact in relationships between organizational groups, and that the attainment of mutual trust leads to shared knowledge. Ribiere (2005) asserts that without trust, individuals will not be likely to share and collaborate in knowledge exchanges. If software developers from different groups trust each other, they are likely to share their knowledge with developers in other groups without worrying that they will be taken advantage of by them. Therefore, we conclude
Hypothesis 3a (H3a): Mutual trust is positively associated with explicit knowledge sharing of software developer dyads from interacting teams.
Hypothesis 3b (H3a): Mutual trust is positively associated with implicit knowledge sharing of software developer dyads from interacting teams.

Project Commitment (PC)

Project commitment can be characterized by the acceptance of and the strong belief in the goals and values of the project, the willingness to engage in the project, and the desire to

maintain membership in the project (Hoegl et al. 2004). With the commitment, a shared, superordinate goal will be achieved. Superordinate goals are important needs shared by members of different groups that can be met only by mutual cooperation (Brewer 2003). Researchers have amassed considerable evidence in support of shared superordinate goals as a mechanism for reducing ingroup favoritism. It is generally assumed that a superordinate goals reduces own group favoritism by bringing outgroup members under the umbrella of a higher-level, shared social identity. With project commitment, the group members involved understand how their work fits into the whole project picture and they are able to set goals that are aligned with the whole project's requirements and deadlines. In terms of social identity theory, ingroup favoritism will be reduced. For the success of the whole project, both implicit and explicit knowledge will be shared more willingly with the outgroup members. Prior studies also propose that a shared goal can be viewed as a bonding mechanism that helps different parts of a network integrate knowledge; inversely, contradicting or inconsistent goals may result in conflict that is not conductive to the flow of knowledge (Inkpen et al. 2005). The superordinate goals may also encourage the development of trusting relationships. In terms of social identity theory, outgroups are usually viewed with suspicion and expected to discriminate against the ingroup. However prior studies suggest that the extent to which a business unit shares a vision with other units and with the organization as a whole will be positively associated with the level of its perceived trustworthiness (Tsai et al. 1998). In conclusion, we propose

Hypothesis 4a (H4a): Project commitment is positively associated with explicit knowledge sharing of software developer dyads from interacting teams

Hypothesis 4b (H4b): Project commitment is positively associated with implicit knowledge sharing of software developer dyads from interacting teams

Hypothesis 4c (H4c): Project commitment is positively associated with mutual trust of software developer dyads from interacting teams

Coordination Technology Use (CT)

Based on previous studies, coordination technology use is defined as functionality of information technologies that enables or supports the interactions of two software developers from interacting teams in the execution of cross-team tasks (Guinan et al. 1998), (Henderson et al. 1990). There are a lot of different kinds of coordination technology available for software development. They cover a serial of functions: Software configuration management, project status, notification services, project scheduling and tasking, CASE and process management, programming tools, bug and change tracking, team memory & knowledge center (Carmel 1999). These functions are effective for coordination. In sum, we propose

Hypothesis 5 (H5): Coordination technology use is positively associated with coordination effectiveness of software developer dyads from interacting teams.

Methodology

Operationalization of Constructs

A two-part survey instrument was designed to get information about all of the variables. Wherever possible, existing scales were used or adapted to enhance validity (Hewstone et al. 2002). Elsewhere, new questions were developed based on a review of the preview literature. All constructs were measured through seven-point scales anchored from "strongly disagree" to "strongly agree" or "never" to "very frequently". A summary of operationalization of constructs is listed in Table 1. Previous studies suggest project characteristics may have an impact on coordination (Kraut et al. 1995). In this study, task certainty, task duration, and task complexity are included as control variables. Software type, which may affect the productivity (Maxwell 1996), also considered as a control variable. In addition, company size

is also controlled since the productivity of software development is usually higher in large company.

Table 1: Operationalization of Constructs

Item Wording and Code	Source
Coordination Effectiveness (CE)	
• Duplicated and overlapping activities were avoided (CE1)	• Hoegl et al. (2004)
• There were no problems in coordinating with each other(CE2)	• Hoegl et al. (2004)
• Conflicts with each other were settled quickly(CE3)	• Hoegl et al. (2004)
• A lot of integration problems were associated with software interfaces (CE4)	• Espinosa (2002)
• Connected processes and activities were well coordinated with each other (CE5)	• Hoegl et al. (2004)
• Discussions with each other were conducted constructively (CE6)	• Hoegl et al. (2004)
Task Interdependence (TI)	
• It was rarely required to obtain information from each other to complete this task (TI1)	• Pearce et al. (1992)
• This task was performed fairly independently of each other (TI2)	
• This task required frequent coordination efforts between us (TI3)	
• Performance on this task was dependent on receiving accurate information from each other (TI4)	
• This task was affected significantly by the way one perform his/her individual job (TI5)	
• This task was planned with little coordination efforts between us (TI6)	
Explicit Knowledge Sharing (EKS)	
• We shared minutes of meetings or discussion records in an effective way (EKS1)	• Bock et al. (2005)
• We always provided technical documents, including manuals, books, training materials, to each other (EKS2)	
• We shared project plans, project status in an effective way (EKS3)	
Implicit Knowledge Sharing (IKS)	
• We always provided know-where or know-whom to each other in an effective way (IKS1)	• Bock et al. (2005)
• We tried to share expertise from education or training in an effective way (IKS2)	
• We always shared experience or know-how from work (including computational, application, domain, and software engineering knowledge) in a responsive and effective way (IKS3)	
Coordination Technology (CT)	
• Access a database, dictionary, diagram, etc. at the same time as another user (CT1)	• Guinan et al. (1998)
• Automatically maintain a record of the bugs and changes (CT2)	• Guinan et al. (1998)
• Instruct the tools to freeze a portion of the design to protect it from changes (CT3)	• Guinan et al. (1998)
• Support notification services (e.g. Notify somebody to do something by automatically sending an email) (CT4)	• Carmel (1999)
• Support software requirement tracking (CT5)	• CMM (1991)
• Send messages to each other (CT6)	• Guinan et al. (1998)
Mutual Trust (MT)	
• I assumed that he or she would always look out for my interests (MT1)	• Levin et al. (2005)
• I felt like he or she cared what happened to me (MT2)	
• I assumed that he or she would go out of his or her way to make sure I was not damaged or harmed (MT3)	
• Given his or her track record, I saw no reason to doubt this person's competence and preparation (MT4)	
• I believed that this person approached his or her job with professionalism and dedication (MT5)	
Project Commitment (PC)	

• We were committed not only to our own teams, but to the overall project (PC1) • We felt fully responsible for achieving the common project goals (PC2) • This project had the strong commitment of us (PC3) • We were proud to be part of the overall project (PC4) • We valued to be part of the overall project (PC5)	• Hoegl et al. (2004)

Survey Administration

A noteworthy point of this study is the way to collect data. Since the dependent variable (coordination effectiveness) is to assess the coordination outcomes of software developer dyads from interacting teams, it is at dyadic level. To avoid the drawback of getting dyadic information from one side of a dyad (one-sided view bias), we developed match-pair survey instruments for software developers. To avoid common method bias (Podsakoff et al. 2003), we developed instrument for managers to evaluate dependent variable. Software developer dyads from interacting teams are required to report on the process variables (knowledge sharing, technology use, mutual trust, project commitment). Their responses were combined to produce one measure of each. Team leaders are required to report on the coordination effectiveness of software developer dyads. A backward translation was used to ensure consistency between the Chinese and the original English version of the instrument (Mullen 1995). Furthermore, two software developers and a manager from China were asked to make any comments this and refine the translations as necessary. The field study was conducted in Mainland China, over a period of about one month, from beginning of March to beginning of April, 2006. Out of 118 software developers, only 16 were female (13.6%). Most of the software developers were in the age group of 25 to 29 years (75.4%). On average, software developers had software development experience of 3.53 years (the range was from 1 to 10 years). All of them were well-educated, 53.4% had bachelor degrees; 46.6% had master degrees. 76.3% of the software developers were engaged in embedded software development (software for network devices). 23.7% were engaged in ERP or office automation systems development. 79.7% of them worked in CMM certificated companies, from level 2 to 4.

Results

Measurement Analysis

The descriptive statistics of the variables in this study is presented in Table 2.

Table 2: Descriptive Statistics

	Mean	SD	1	2	3	4	5	6	7
1. CT	4.18	1.36	**0.844**						
2. EKS	5.47	.92	0.299	**0.884**					
3. IKS	5.28	.85	0.23	0.563	**0.812**				
4. TI	5.26	.74	0.042	0.306	0.377	**0.794**			
5. MT	5.14	.70	0.17	0.456	0.526	0.549	**0.792**		
6. PC	5.63	.82	0.118	0.506	0.446	0.573	0.461	**0.886**	
7. CE	4.81	.92	0.147	0.355	0.507	0.682	0.557	0.427	**0.722**

Individual Level Reliability: We assessed reliabilities of all independent variables by calculating Cronbach's alpha at individual level . All the Cronbach's alpha values were found to be greater than 0.7, the threshold suggested by Nunnally (1978).

Individual Level Convergent and Discriminant Validity: In order to assess the convergent and divergent validity of the scales, a factor analysis with principal components analysis and varimax rotation was used. One question for explicit knowledge sharing (EKS2) and one question for task interdependence (TI2) tapped into other constructs and were omitted. All

other items had at least good loadings on their intended constructs (the minimum loading is 0.552). These 6 factors that emerged explained about 68.2% of the total variance.

Aggregation Analysis: For the independent variables, they were collected at individual level. We used two procedures to statistically test the conformity of the level of measurement to the level of the theoretical analysis, called 1) ICC, which compares within-group variation to between-group variation, and 2) Rwg, which access within-team rater agreement (Grawitch et al. 2004). All the F values are significant. We can conclude that the dyadic level analysis is more appropriate than individual level analysis. For the variables to be aggregated by average, the average Rwg values are all above the cut-off value 0.7. This reflects a high degree of within-group agreement. Table 3 provides the Rwg, ICC and F values for this study's variables.

Table 3: Summary of ICC and Rwg

Construct	ICC	*F* value	p-value	R_{wg}
CT	0.537	3.33	< 0.01	0.703
EKS	0.374	2.19	< 0.01	0.854
IKS	0.253	1.65	< 0.05	0.847
TI	0.247	1.65	< 0.05	0.811
PC	0.460	2.67	< 0.01	0.927

Dyadic Level Convergent and Discriminant Validity: Three tests were used to assess convergent validity: loadings of items, composite reliability of constructs, and average variance extracted by constructs (Fornell 1981). Table 4 presents the results of convergent validity tests. We can see that all the loadings are larger than 0.5 and t-values for these loadings are above 1.96 (0.5 indicates adequate reliability). Composite reliability values range from 0.811 to 0.944 (0.8 indicates adequate composite reliability). The average variances extracted by the measures range from 0.521 to 0.785 (0.5 indicates acceptability (Fornell 1981)), which are above the acceptability value. Therefore, we can conclude that all constructs in this study have adequate convergent validity. Two tests were conducted to assess discriminant validity: (1) examining item loadings to construct correlation. (2) examining the ratio of the square root of the average variance extracted of each construct to the correlations of this construct to all the other constructs (Gefen et al. 2005). Table 4 presents the loading and cross-loading of all remaining items. The average variance extracted for PC was calculated again as 0.785. The square roots of the average variance extracted for each construct are greater than the levels of correlations involving the construct. In sum, the results from these two tests confirm the discriminant validity.

Table 4: Results of Convergent Validity Tests and Factor Analysis

Construct	Items	t-value	Composite Reliability	Average Variance Extracted	1	2	3	4	5	6	7
1. CT	CT1	3.48	0.908	0.713	**0.80**	0.26	0.21	0.03	0.15	0.11	0.09
	CT2	3.83			**0.90**	0.29	0.18	0.00	0.14	0.03	0.11
	CT3	4.17			**0.92**	0.23	0.21	0.07	0.14	0.15	0.14
	CT5	2.71			**0.75**	0.14	0.15	0.00	0.04	0.01	-.03
2. EKS	EKS1	13.79	0.878	0.780	0.12	**0.84**	0.37	0.29	0.34	0.38	0.21
	EKS3	35.06			0.37	**0.93**	0.59	0.26	0.45	0.50	0.39
3. IKS	IKS1	13.78	0.853	0.660	0.25	0.62	**0.86**	0.37	0.48	0.57	0.51
	IKS2	18.15			0.11	0.36	**0.84**	0.24	0.33	0.25	0.36
	IKS3	7.20			0.16	0.31	**0.74**	0.28	0.45	0.17	0.32
4. TI	TI2	22.78	0.834	0.631	0.07	0.27	0.37	**0.88**	0.49	0.51	0.63
	TI3	19.24			0.01	0.41	0.33	**0.85**	0.53	0.64	0.60
	TI4	5.70			0.01	-0.1	0.15	**0.63**	0.21	0.08	0.33

5. MT	MT1	5.48	0.892	0.627	0.18	0.24	0.36	0.24	**0.64**	0.17	0.32
	MT2	26.80			0.19	0.40	0.62	0.48	**0.87**	0.36	0.59
	MT3	22.05			0.14	0.39	0.42	0.41	**0.85**	0.30	0.38
	MT4	20.74			0.17	0.39	0.34	0.50	**0.84**	0.49	0.45
	MT5	9.27			-.02	0.35	0.30	0.50	**0.74**	0.46	0.40
6. PC	PC1	24.23	0.944	0.785	0.04	0.44	0.32	0.54	0.39	**0.90**	0.36
	PC3	27.55			0.10	0.54	0.47	0.46	0.39	**0.84**	0.40
	PC4	22.68			0.15	0.39	0.40	0.56	0.47	**0.91**	0.40
	PC5	26.02			0.12	0.40	0.37	0.46	0.38	**0.90**	0.35
7. CE	CE1	8.63	0.811	0.521	0.14	0.35	0.38	0.51	0.26	0.30	**0.65**
	CE2	6.76			0.14	0.22	0.41	0.35	0.49	0.28	**0.68**
	CE3	5.21			0.01	0.15	0.17	0.53	0.40	0.23	**0.69**

Structure Model Assessment

With an adequate measurement model, the proposed hypotheses were tested with PLS version 3.0. The results of the analysis are presented in Figure 2.

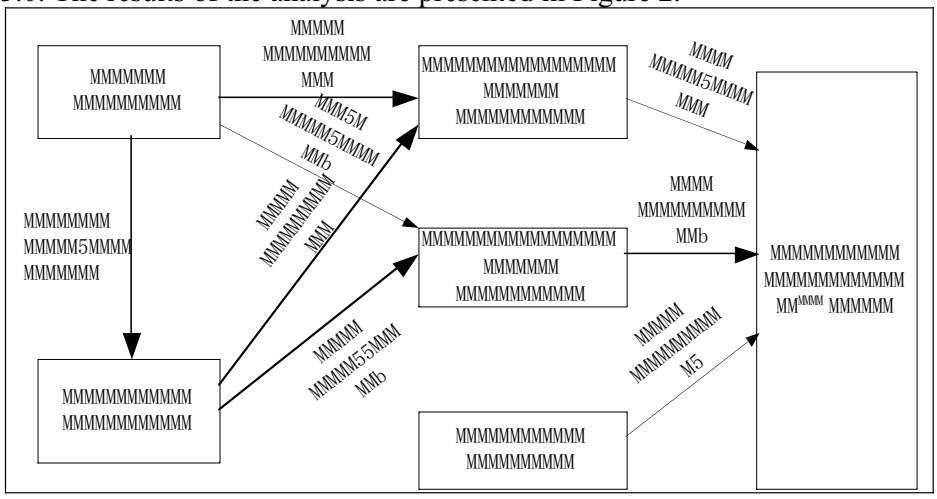

Figure 2: Result of PLS Analysis

As shown in Figure 2, 29.2 percent of the variance in coordination effectiveness is explained (10% is an indication of substantive explanatory power). Control variables were included in the model. For project complexity, project duration, and project uncertainty, the aggregated values by average were used. A categorical variable was created to measure software type (embedded software and others). The number of software developers in respondents' companies was also included based on the categorical values. The t-value is 0.7334 for project complexity, 1.5783 for project duration, 1.5983 for project uncertainty, 0.5422 for software type, and 0.3973 for the number of software developer. None of these control variables were found to be significant.

As hypothesized, implicit knowledge sharing is significant associated with coordination effectiveness (path coefficient = 0.48, t-value = 4.1401, $p < 0.01$), supporting H1b. However, explicit knowledge sharing has no significant effect on coordination effectiveness (path coefficient = 0.09, t-value = 0.5923, p-value = 0.5536 > 0.1). Thus, hypotheses H1a is not supported. Since explicit knowledge sharing does not have a significant impact on coordination effectiveness. We can conclude H2a is not supported. The moderating effect of task interdependence on the relationship between explicit knowledge sharing and coordination effectiveness does not exist. H2b are not supported. Mutual trust has a significant positive impact on both explicit knowledge sharing (path coefficient = 0.282, t-

value = 2.3608, p-value = 0.0182 < 0.05) and implicit knowledge sharing (path coefficient = 0.408, t-value = 2.5543, p-value = 0.0106 < 0.05), supporting H3a and H3b. Project commitment has a significant impact on explicit knowledge sharing (path coefficient = 0.376, t-value = 2.4938, p-value = 0.0126 < 0.05) and mutual trust (path coefficient = 0.461, t-value = 3.5023, p-value < 0.01), supporting H4a and H4c.However, it doesn't have a significant impact on implicit knowledge sharing (path coefficient = 0.256, t-value = 1.5033, p-value = 0.1328 > 0.1).Thus, H4b is not supported. Coordination technology has no significant impact on coordination effectiveness (path coefficient = 0.017, t-value = 0.1102, p-value = 0.9123 > 0.1), Thus, H5 is not supported.

Discussion

Based on the results, explicit knowledge sharing did not significantly affect coordination effectiveness of software developer dyads from interacting teams. There are two possible interpretations for this result. First, this may be due to the fact that software developers can easily get related explicit knowledge, such as minutes of meetings or discussion records, project plans, and project status, from other sources. The second possible reason is that explicit knowledge may only have limited impact on coordination effectiveness. Since software development is quite complex, it is not sufficient to achieve coordination effectiveness only by explicit knowledge. As hypothesized, implicit knowledge sharing had a significant positive impact on coordination effectiveness of software developer dyads from interacting teams. Due to the thin spread domain knowledge (Curtis et al. 1988), there are so few "project gurus" who have a thorough comprehension of the whole system in large scale software projects. It is not likely for a software developer to know much about the details of the subsystems or modules developed by an outgroup member. When issues, such as interfaces, debugging information, occur, unlike explicit knowledge they may get from other sources, software developers often have to consult with their counterparts for their expertise.

The relationship between coordination technology use and coordination effectiveness was not statistically significant. This result is a little bit contrary to commonly accepted ideas. Such a finding might simply be a reflection of the usage of coordination technology in the sampled software development projects. In these projects, software developers were collocated. They could easily meet each other face to face. E-mail was the most frequently used IT tool. Therefore, the role of coordination technology might be not salient. Further, due to the small sample size (59) and relatively high standard deviation of coordination technology (1.36), the power might be impacted. However, several previous studies also have similar findings. Some researchers argue that it may be due to the lack of coordination functionality inherent in many CASE tools (Guinan et al. 1998; Henderson et al. 1990; Vessey et al. 1995). Sawyer et al. argue that tools do help software developers improve individual productivity but they are not directly linked to team performance (1998). According to them, tools usage may have the unintended effects of guiding valued social processes. The impact of coordination technology deserves more future studies.

The moderating impact of task interdependence on the relationship between both explicit and implicit knowledge sharing was not supported as hypothesized. There are several possible reasons for this. First, moderating effects require greater power to detect than do main effects. Since task interdependence is relatively high and homogeneous in our sample (mean is 5.2655, standard deviation is 0.74). In high interdependence contexts, it is necessary to share implicit knowledge for better coordination. Thus, moderating effect can not be supported. Alternatively, it might be the case that implicit knowledge sharing always predicts coordination effectiveness, independent of the situational contingency of task

interdependence. Therefore, this moderating effect warrants further empirical examination. More heterogeneous samples in respect to task interdependence are needed for further studies.

The relationship between mutual trust and explicit knowledge sharing as well as implicit knowledge sharing was significant. As hypothesized, when mutual trust between software developer dyads is high, they tend to be more motivated to share both explicit and implicit knowledge. This result is consistent with previous studies (Tsai et al. 1998), (Levin et al. 2004). Project commitment was found significantly related to explicit knowledge sharing. However, it was not significantly directly related to implicit knowledge sharing. The impact was mediated by mutual trust. It appears that it is not sufficient to share implicit knowledge with only project commitment. This result seems to be reasonable. It is well understood that implicit knowledge is more difficult and time-consuming to articulate and transfer than explicit knowledge (Nonaka et al. 1995). To share implicit knowledge, the quality of the relationship between a knowledge seeker and a knowledge source is critical (Simonin 1999). But members may embrace the same organization goals and values even they do not have a good interpersonal relationship. Compared with mutual trust, project commitment does not strongly reflect relationship quality. Previous studies also find that the relationship between shard vision and resource exchange is mediated by trust (Tsai et al. 1998).

Implications and Conclusion

This study advances theoretical development in the area of coordination research in software development. To our best knowledge, it is the first time to investigate antecedents of cross-team coordination effectiveness in software development. As discussed before, cross-team coordination is very important for some system-level issues in software development. This study provides some insights into the team-external processes by examining coordination of software developer dyads from interacting teams. Future studies can be conducted more deeply based on the findings of this study.

This study investigates the roles of social capital and coordination technology in coordination effectiveness of software dyads from interacting team. The results suggest that social capital plays an important role, and that the role of coordination technology is marginal. Software engineering literature tends to overemphasize the contribution of technical solutions. Unfortunately, technical approaches are not the “silver bullet”. Consistent with previous studies in IS research (Guinan et al. 1998; Henderson et al. 1990; Vessey et al. 1995), our study suggests more attention should be paid to understand the social processes or behavior in software development. This also conforms to the spirits of agile software development methods (Cockburn 2002), which place the emphasis on human factors.

In addition, this study examines knowledge sharing between software developer dyads from interacting teams. It also contributes to the area of knowledge management in general, especially to knowledge sharing research in software development. As proposed, research on how properties of relationships affect knowledge management is very important and promising since relationships are critical when one moves beyond studying individuals to studying social units (Argote 2003). This study explores the impact of some relationship properties, such as mutual trust and project commitment, on explicit and implicit knowledge sharing in the context of software development. Most noteworthy, this exploration is conducted from a dyadic perspective. Compared with prior studies which get information from only one side of a relationship, it has obvious advantage.

References

Adler, P.S., and Kwon, S.W. "Social Capital: Prospects for A New Concept," *Academy of Management Review* (27:1) 2002, pp 17-40.

Ancona, D.G. "Outward bound: Strategies for Team Survival in an Organization," *Academy of Management Journal* (23:2) 1990, pp 334-365.

Brewer, M.B. *Intergroup Relations* Open University Press, Buckingham. Philadephia, 2003.

Brown, R. "Social Identity Theory: Past Achievements, Current Problems, and Future Challenges," *European Journal of Social Psychology* (30:6) 2000, pp 745-778.

Carmel, E. *Global Software Teams: Collaborating Across Borders And Time Zones* Prentice Hall PTR, 1999.

Cockburn, A. *Agile Software Development* Addison-Wesley, Boston, 2002.

Crowston, K., and Kammerer, E.E. "Coordination and Collective Mind in Software Requirments Development," *IBM Systems Journal* (37:2) 1998, pp 227-245.

Cummings, L.L., and Bromiley, P. "The Organization Trust Inventory: Development and Validation," in: *Trust in Organizations: Frontiers of Theory and Research,* R.M. Kramer and T.R. Tyler (eds.), Sage, Thousand Oaks, CA, 1996.

Curtis, B., Krasner, H., and Iscoe, N. "A Field Study of the Software Design Process for Large Systems," *Communications of the ACM* (31:11) 1988, pp 1268-1287.

Espinosa, J.A. "Shared Mental Models and Coordination in Large-scale, Distributed Software Development," in: *Graduate School of Industrial Administration (Information Systems)*, Carnegie Mellon University, 2002, p. 130.

Fenema, P.C.v. "Coordination and Control of Globally Distributed Software Projects," in: *Erasmus Research Institute of Management*, Erasmus University, 2002, p. 574.

Gefen, D., and Straub, D. "A Practical Guide to Factorial Validity Using PLS-Graph: Tutorial and Annotated Example," *Communications of the Association for Information Systems* (16:5) 2005, pp 91-109.

Grawitch, M.J., and Munz, D.C. "Are Your Data Nonindependent? A Practical Guide to Evaluating Nonindependent and Within-Group Agreement," *Understanding Statistics* (3:4) 2004, pp 231-257.

Guinan, P.J., Cooprider, J.G., and Faraj, S. "Enabling Software Development Team Performance During Requirements Definition: A Behavioral Versus Technical Approach," *Information Systems Research* (9:2) 1998, pp 101-125.

Henderson, J.C., and Cooprider, J.G. "Dimensions of I/S Planning and Design Technology," *Information Systems Research* (1:3) 1990, pp 227-254.

Hewstone, M., Rubin, M., and Willis, H. "Intergroup Bias," *Annual Review of Psychology* (53:1) 2002, pp 575-604.

Hoegl, M., Weinkauf, K., and Gemuenden, H.G. "Interteam Coordination, Project Commitment, and Teamwork in Multiteam R&D Projects: A Longitudinal Study,") 2004.

Inkpen, A.C., and Tsang, E.W.K. "Social Capital, Networks, and Knowledge Transfer," *Academy of Management Review* (30) 2005, pp 146-165.

Kane, A.A., Argote, L., and Levine, J.M. "Knowledge Transfer between Groups via Personnel Rotation: Effects of Social Identity and Knowledge Quality," *Organizational Behavior and Human Decision Processes* (96:1) 2005, pp 51-71.

Kotlarsky, J., and Oshri, I. "Social Ties, Knowledge Sharing and Successful Collaboration in Globally Distributed System Development Projects," *European Journal of Information Systems* (14:1) 2005, pp 37-48.

Kraut, R.E., and Streeter, L.A. "Coordination in Software Development," *Communications of the ACM* (38:3) 1995, pp 69-81.

Levin, D.Z., and Cross, R. "The Strength of Weak Ties You Can Trust: The Mediating Role of Trust in Effective Knowledge Transfer," *Management Science* (50:11) 2004, pp 1477-1490.

Malone, T.W., and Crowston, K. "The Interdisciplinary Study of Coordination," *ACM Computing Surveys* (26) 1994, pp 87-119.

Mullen, M.R. "Diagnosing Measurement Equivalent in Cross-National Research," *Journal of International Business Studies* (26:3) 1995, pp 573-596.

Nonaka, I., and Takeuchi, H. *The Knowledge-Creating Company* Oxford University Press, New York,

1995.

Paulk, M.C., Curtis, B., Chrissis, M.B., and Webber, C. "Capability Maturity Model for Software ver. 1.1," Technical Report CMU/SEI-93-TR-24, Software Engineering Institute, Pittsburgh, PA.

Podsakoff, P.M., MacKenzie, S.B., Lee, J.Y., and Podsakoff, N.P. "Common Method Biases in Behavioral Research: A Critical Review of the Literature and Recommended Remedies," *Journal of Applied Psychology* (88:5) 2003, pp 879-903.

Sawyer, S., and Guinan, P.J. "Software Development: Processes and Performance," *IBM Systems Journal* (37:4) 1998, pp 552-568.

Simonin, B. "Ambiguity and the Process of Knowledge Transfer in Strategic Alliances," *Strategic Management Journal* (20:7) 1999, pp 595-623.

Tajfel, H. *Human Groups and Social Categories* Cambridge Univeristy Press, Cambridge, 1981.

Tajfel, H., and Turner, J.C. "The Social Identity Theory of Intergroup Behavior," in: *Psychology of Intergroup Relations,* S. Worchel and W. Austin (eds.), Nelson-Hall, Chicago, 1986.

Tsai, W., and Ghoshal, S. "Social Capital and Value Creation: the Role of Intrafirm Networks," *Academy of Management Journal* (41) 1998, pp 464-476.

Turner, J.C. "Social Identification and Psychological Group Formation," in: *The Social Dimension: European Developments in Social Psychology, Vol.2,* H. Tajfel (ed.), Cambridge University Press, Cambridge, 1984.

Tushman, M.L., and Nadler, D.A. "Information Processing as An Integration Concept in Organizational Design," *Academy of Management Review* (3) 1978, pp 613-624.

Vessey, I., and Sravanapudi, A.P. "CASE Tools as Collaborative Support Technologies," *Communication of ACM* (30:1) 1995, pp 83-95.

Weinberg, G.M. *The Psychology of Computer Programming: Silver Anniversary Edition* Dorset House Publishing, 1998.

Wittenbaum, G.M., and Stasser, G. "Management of Information in Small Groups," in: *What's Social about Social Cognition?Research on Socially Shared Cognition in Small Groups,* J.L. Nye and A.M. Brower (eds.), Sage Publication, London, 1996.