Knowledge Transfer in Non-Collocated Software Architecture Development: From the Perspective of Analysts and Software Architects

Salfarina, A.[1], Marzanah, A. J.[2], Sazly, A.[3]

Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Malaysia
[1]sal79.sa@gmail.com, [2]marzanah@putra.upm.edu.my, [3]sazly@fsktm.upm.edu.my

ABSTRACT

Learning within software development involves the transfer of knowledge between different yet interdependent functional teams. In reality however, these teams often create islands of knowledge due mostly to indistinct flow of knowledge transfer (KT), thus fail to take advantage of the opportunity to learn from each other. Taking the non-collocated software architecture development teams as a challenge, the goal of this study is to investigate the nature of KT that occurs between the analyst and software architect teams in non-collocated software architecture development. Data are collected from semi-structured interviews with 30 respondents consisting of industrial experts ranging from analyst, software architects and project managers. We managed to gather sufficient evidence that proves KT occurs, and successfully characterize the areas of knowledge used and exchanged, the interdependency between teams, the utilization of knowledge, the medium used for KT and finally, the external conditions surrounding KT during non-collocated software architecture development. These findings are useful as they rest a good understanding of KT and its vital elements in non-collocated software architecture development for all prospective researchers and practitioners.

Keywords: Knowledge transfer (KT), non-collocated software architecture development, analyst, software architects.

I INTRODUCTION

Literature review indicates that there is KT in software development. Software architecture development in particular, is highly recognized as a phase where knowledge integration mostly occurs to determine the outcome of subsequent development processes. The encounter between analyst and software architect teams as the prominent roles in developing software architecture have highlighted the need for KT in order to help accelerate and better facilitate each teams' responsibility towards completing their tasks. However, KT between non-collocated teams is often problematic. One of the biggest issues is lack of understanding of the process. In other words, having inadequate details on how the knowledge is being transferred, from whom and to whom, including the content of the knowledge and how it will be made into use. Several studies have proven that within software development, KT occurs more often, informally. Therefore our study aims to provide a complete picture that acts as a guideline of reference for prospective researchers and practitioners about the essentials of KT. In the next sections, the methodology is briefly highlighted, the results and discussions are explained, followed by the conclusions of the results.

II METHODOLOGY.

Each interview session is done individually at the respondent's preferred location. The interviewer was the researcher herself and assisted by a research assistant. Since the interview exercised the semi structured form of questionnaires, every session took at least 1 hour to complete.

III RESULTS AND DISCUSSIONS

We adopt a communication-based perspective and the orientation of knowledge flow (Jablin and Putnam, 2001; Szulanski, 2000; Wei'e, 2011) that has often been used to study virtual or distributed teams, which indicates five basic elements that determine and influence the transfer of knowledge: channel, message, context, recipient, and source. In addition, there are elements called the evaluation (Berlo, 1960; Jablin & Putnam, 2001) and external environmental (Wei'e, 2011) that have also been claimed to influence KT. In what follows, we simultaneously present and discuss our findings drawn from those key elements of KT.

A. The Areas of Knowledge Used and Exchanged

To address the first factor, a list of knowledge areas identified from the literature as being most relevant and significant to both analyst and software architect were initially investigated. The list comprises of four distinct areas of knowledge including technical, application domain, project management and people knowledge. Technical knowledge area encompasses a breadth of knowledge; programming, problem solving strategies, code testing & debugging, development knowledge and skills, architecture concepts & techniques, detailed design, design constraints, specific and general technologies & platforms, software development methods and specification techniques & languages, software design principles, abstractions of design/code as schemas or plans, and design techniques & tools (Harandi, 1998; Joshi et al., 2007; Faraj & Sproull, 2000; Ko et al., 2005; Rus & Lindvall, 2002; Walz et al., 1993 ; Hansen, 2002 ; Convoy & Soltan, 1998 ; Boloix & Robillard, 1995; Ramesh & Tiwana, 1999; Correa, 1996; Tiwana, 2004). Application domain knowledge area concerns about the specific system to which the software pertains, customers' business process, client operations, business rules, stakeholders' needs, as well as the customers' business objectives (Harandi, 1998; Faraj & Sproull, 2000; Rus & Lindvall, 2002; Convoy & Soltan, 1998; Walz et al., 1993; Boloix & Robillard, 1995).While project management knowledge deals with planning, staffing, managing and leading a project (Ko et al., 2005; Rus & Lindvall, 2002; Correa, 1996), people knowledge on the other hand, accounts the knowledge about leadership, teamwork, communication, negotiation, accepting direction, mentoring and consulting (Bass et al., 2008). Table 1 summarizes the frequency of agreement of both teams pertaining to each knowledge area.

Table 1. Results regarding knowledge areas as perceived important to both analyst and software architect

Knowledge areas	Frequency of agreement (YES or NO)	Percent (%)
Technical	30 –YES	100
Application domain	30 – YES	100
Project Mgt.	20 – YES 10 – NO	66.7 33.3
People	30 – YES	100

All 30 participants unanimously believe that technical, application domain and people knowledge areas are valuably important for them to complete their tasks. Surprisingly, only 20 participants perceive that project management knowledge area is useful during the development of software architecture. The other 10 participants who believe otherwise might partly be driven by the thought that planning, staffing, managing or scheduling timeline or the project as a whole is not their primary responsibility. One participant gave a similar comment when asked why he does not perceive project management as equally important:

> *"We have project manager and team leader to deal with these kinds of stuff. It's an advantage to know some about managing project but we prefer to focus in our real tasks."*

Then the participants were asked an open question about three topics or specific areas of knowledge that are most commonly exchanged and discussed between teams during software architecture development. Table 2 illustrates a compilation of their responses. We analyze these responses by categorizing the specific topics accordingly to the knowledge areas. We have found that most of the topics exchanged between the two teams are mainly based on the deliverables and discussion activity during the process of software architecture development itself. The deliverables are typically in the form of documentation artifacts. Topics discussed during the process of software architecture development are generally about making negotiations regarding the requirements, managing clients' expectations as well as explaining rationales of the design. They also share about each other's experience from working in previous projects.

The areas of technical and application domain knowledge were the most commonly exchanged and discussed between both analyst and software architect teams. Their dominance implies that in developing software architecture, the integration of technical and application domain knowledge is a must to ensure completion of the given tasks to produce desired deliverables. This is further supported by Tiwana (2004) and Faraj & Sproull (2000), who state that in devising a coherent software solution (software architecture) for a business problem, these two areas of knowledge are germane to the process. Some participants also stress the ultimate importance of technical and

application domain knowledge areas by saying that:

"Notwithstanding the importance of other knowledge areas, we do rely heavily upon the technical and application domain knowledge in accomplishing our tasks"

Table 2. Specific topics/areas of knowledge

Knowledge Areas	**Specific topics/areas of knowledge**
Technical	Use case diagram – overall system flow
	DFD, ERD
	System specification
	Component diagram architecture
	Standards
	Architectural principals and rules
	Technical constraints
	Detailed design specifications
	Design decisions
	Documentation: BRD, SDP, SRS, SDD, FRD, TRS. FRS
Application Domain	Business process prototypes
	Business rules for business process
	Domain subjects
	Business model
	Functional and non functional requirement
Project Management	Gantt Chart – due date of completion
	Assignment delegation among team members
	Ad-hoc meeting scheduling
People	Rational trade-off concerning the requirements, technical constraints
	Client's expectations & priorities negotiations
	Past experiences from working on other projects
	Communicating the deliverables

The results depicted from Table 2 also do not contradict with our prior postulation regarding the areas of knowledge exchanged and used during software architecture development. In fact, we can conclude that the transfer of knowledge during the development of software architecture is mainly stemmed from these four areas of knowledge as indicated specifically in the table.

B. The Interdependencies between Teams

Despite of physical dispersion, the necessity to share and exchange knowledge between teams is continuously stimulated by the need to produce the desired deliverables from one phase to another. "In software development, teams are often highly dependent on one-another and that the dependencies are not sequential …which means the two teams work closely together…" (Sawyer, 2001). Additionally, requirement management and architectural design evolve in parallel and support each other (Kruchten, 2011). This has lending further support as to display the interdependencies that exist between both analyst and software architect teams although are non-collocated.

Based from the interviews, we learnt that the interdependencies between these non-collocated teams stem from the task and team interdependencies. Task interdependencies in general refer to the extent to which one group is dependent upon one another to perform their tasks. They have to gain as much input as required to perform and complete the given tasks. This extends to the interdependencies explained by the necessity to access other expertise located in another team in order to carry out the assignments. Existing studies have provided ample evidence that both collocated and distributed software development teams frequently engage in communication to acquire necessary information from peer developers (Ko et al. 2007, La Toza et al. 2006). In this case, both teams play the role of both knowledge sender and receiver.

Although each team seems totally foreign to each other in terms of the skills and expertise, they actually share a lot of traits. Both teams deal with making decisions as well as relying more on the experiences. The overlapping picture displayed by the nature of their tasks has induced stronger support to confirm that there are serious interdependencies between the teams.

Concerning the questions in regards to the interdependencies between teams, all of the participants are in agreement that the interdependencies exist between both teams are primarily driven by the several highlighted reasons. Firstly is to gain as much input as required to complete the tasks and produce desired deliverables. Secondly, is to obtain knowledge and understanding of a particular aspect of the software artifact under investigation. Third, is to gain access to expert for their valuable experiences and knowledge obtained from previous projects. La Toza et al. (2006) describe the role of the "*team historian*" who possesses knowledge about the origins of a project and its architecture", in which this kind of knowledge is not obtainable from any artifacts resources. And fourthly, is to coordinate development activities among them.

C. The Utilization of Exchanged Knowledge

In this study, our interest lies in determining the extent of knowledge utilization during software architecture development among both analyst and software architect teams. Our strategy was to list down 23 items concerning the application of related knowledge into each possible step-by-step activity in software architecture development. Every item asked was constructed in a way it tells where the participant gain the knowledge from, and how does the knowledge being put into use to accommodate the activities involved. We name this method as knowledge utilization characterizing.

As anticipated, majority of the participants have successfully characterized the extent of their knowledge utilization. As shown in Table 3, 100% of the participants agree and strongly agree that they perform all of the listed items regarding knowledge utilization. This suggests that they have engaged in KT and prove that they have actually applied the knowledge they gained into their tasks. This is consistent with the requirement or prerequisite of effective KT that emphasizes putting the knowledge into action and not merely knowledge transferring and receiving situation.

We also found that although both teams produce different deliverables, their tasks are overlapping dependent by nature. This simply means that there are tasks involving both teams that rely on their capability to make mutual decision, "… in order to reach a consensus regarding the multiple interpretations of the software requirements … and clarify any existing instances of role ambiguity" (Andres, 2002). They are not just sequentially dependent but they corroborate each other to accomplish their tasks. For example, as commented by one of the participants:

"As a software architect, although I am not directly involved in requirements gathering, I work together with the SA (analyst) to articulate and refine architectural requirements. This is important to ensure that the architecture fulfills the requirements and clients' expectations."

The reason we highlight the existence of overlapping tasks between these teams is to show that despite of distance barrier, both teams still keep themselves engaged in KT.

Table 3. Characterization of knowledge utilization

Items	Frequency (and percentage %)		
	Somehow agree	**Agree**	**Strongly agree**
Using the knowledge gained from the mentoring session held prior to starting the project, we analyze software requirements.	0 (0%)	28 (93.3%)	2 (6.7%)
We held regular meetings and discussions for both teams in order to ensure we understand business and customer needs before development begins.	0 (0%)	26 (86.7%)	4 (13.3%)
We capture software specifications from business requirements described by the clients through brainstorming session.	0 (0%)	21 (70%)	9 (30%)
Using our architectural and design knowledge, we articulate and refine architectural requirements.	11 (36.7%)	18 (60%)	1 (3.3%)
Using our knowledge in software development methods, we document the defined requirements to produce SRS.	0 (0%)	27 (90%)	3 (10%)
Through several meetings and progress reviews, we get input on needs to evolve and improve the architecture.	0 (0%)	28 (93.3%)	2 (6.7%)
We create/draw the initial architecture based on an analysis of the given requirements.	3 (10%)	24 (80%)	3 (10%)
We often use reference architecture and make some adjustments to save time on architectural decisions.	10 (33.3%)	20 (66.7%)	0 (0%)
We make design decisions based on mutual agreement with the other team.	4 (13.3%)	18 (60%)	8 (26.7%)
Using our architectural and design knowledge, we identify the style and articulate the principles and key mechanisms of the architecture partitioning the system.	9 (30%)	16 (53.3%)	5 (16.7%)
We define how the various components fit together.	3 (10%)	27 (90%)	0 (0%)
We evaluate the architecture through various means including prototyping, reviews, and assessments.	5 (16.7%)	25 (83.3%)	0 (0%)
We do trade-off analysis on the design through active discussions with the business/software analyst team.	4 (13.3%)	24 (80%)	2 (6.7%)

Using the application domain knowledge gained from the early phase of requirement analysis, we document the domains for which the system/software will be built.	2 (6.7%)	22 (73.3%)	6 (20%)
We prepare architectural documents and deliver presentations to the stakeholders and other development teams.	0 (0%)	27 (90%)	3 (10%)

Recall that we choose to define KT as learning from the experience of others. It is worth noting that every activity in the software architecture development involves collaboration of both analyst and software architect teams. The task specified for each activity either requires the application of knowledge obtained from previous engagement with other people/team or necessarily demand for participation from other people/team for their input, view and agreement on certain issues. This has therefore strengthened the fact that KT in software architecture development does not only address the utilization of knowledge but put the emphasis in the essentials of learning from others and their experiences. We extend our effort by proving that despite of physical distance, these teams (analysts and software architects) managed to characterize their knowledge utilization which span from technical, application domain, project management to people knowledge throughout the activities involved.

D. The Mediums Used for KT

For dispersed teams, the ideal means for KT are translated through communication technologies. The activities in software architecture development demonstrate such a knowledge intensive environment that not only integrate diverse knowledge, skills and expertise from different group of people but also demand a great deal of communication to ensure the deliverables produced are as expected. More importantly, sufficient efforts need to be addressed to adequately facilitate these dispersed teams in accomplishing their goals. In our study, our attention is directed into determining types of mediums utilized for KT. We provide a list of potential mediums that are used for KT as shown in Table 4 below. The frequency column indicates the number of participants who chose each medium. Email or electronic mail, review meetings and document preparations make the top three lists. This is followed by presentations, training courses, workshops and on-line forums. There are also participants that choose other mediums such as teleconferencing, videoconferencing, face-to-face discussion, social networks, and intranet. These observable findings signify the diversity of mediums used for KT, which implicitly highlights the importance of KT itself. In addition, these findings also suggest their vitality to reduce uncertainty and equivocality associated with the information requirements of the assigned tasks (Andres, 2002). Uncertainty reduction refers to the elimination of the lack of information needed to complete the tasks. Equivocality reduction on the other hand, refers to reducing the ambiguity associated with a task.

Table 4. Result of mediums used for KT

Medium	Frequency	Percentage (%)
Document preparations	30	100
Review Meetings	30	100
Email	30	100
Presentations	28	93.3
Training courses	27	90.0
Workshops	23	76.7
Knowledge portals/discussion forums	19	63.3
Teleconferencing	11	36.7
Face-to-face discussion	9	30.0
Videoconferencing	8	26.7
Social networks	6	20.0
Intranet	4	13.3
Desktop computer conferencing	0	0
Extranet	0	0
Story telling	0	0
Conferences	0	0

The findings also indicate the use of different categories of communication media including lean and rich media. Email, intranets, knowledge portals/online forums, social networks are categorized as lean media. Rich media includes videoconferencing, teleconferencing, face-to-face meetings, training courses, and workshops. Based from the table, we can see that the utilization of rich media dominates over lean media. This is particularly an interesting finding since we are studying non-collocated teams, in which despite of physical constraint, they still manage to meet up face-to-face. One reason that best explain this is most of the knowledge is partly tacit, which is not easily transferred to others. However, we learnt that most of the time, the meetings were done unplanned, or ad-hoc. This usually caused by unexpected demands or changes over the requirements and design that need immediate attention. Architecture evaluation is another cause for such ad-hoc meetings to be organized. As anticipated, during any other times, any problems

or issues arise regarding the tasks assigned between teams are discussed and solved over the phone or emails.

Following the response from the participants regarding the specific topic or areas of knowledge used and exchanged during software architecture development, we provide some extensions as shown in Table 5 below that suggests different mediums, which can be employed for KT according to the nature of knowledge to be transferred. In general, the nature of knowledge can be either categorized as explicit or tacit. As depicted in the table, technical knowledge transferred is predominantly explicit in nature thus calls for lean media to facilitate KT. Explicit knowledge is transferred most efficiently through written media because it will save the unnecessary communication costs associated with face-to-face communication (Pedersen et al., 2003).

Table 5. Suggested medium for KT

Knowledge Areas	**Specific topics/areas of knowledge**	**Nature of knowledge**	**Suggested medium for transfer**
Technical	Use case diagram – overall system flow	Predominantly Explicit	**Lean Media** Examples: *Email Documentation Discussion Forum*
	DFD, ERD		
	System specification		
	Component diagram architecture		
	Standards		
	Architectural principals and rules		
	Technical constraints		
	Detailed design specifications		
	Design decisions		
	Documentation: BRD, SDP, SRS, SDD, FRD, TRS. FRS		
Application Domain	Business process prototypes	Predominantly a combination of Explicit and Tacit	**Lean and Rich Media** Examples: *Email Documentation Training courses Workshops*
	Business rules for business process		
	Domain subjects		
	Business model		
	Functional and non functional requirement		
Project Mgt.	Gantt Chart – due date of completion	Predominantly a combination of Explicit and Tacit	**Lean and Rich Media** Examples: *Email Documentation Review meetings Mentoring*
	Assignment delegation among team members		
	Ad-hoc meeting scheduling		
People	Rational trade-off concerning the requirements, technical constraints	Predominantly Tacit	**Rich Media** Examples: presentation *Face-face discussion Teleconference Videoconference*
	Client's expectations & priorities negotiations		
	Past experiences from working on other projects		
	Communicating the deliverables		

On the other hand, application domain and project management knowledge are mainly comprised of combination of both explicit and tacit, which suggests for the use of lean and rich media. People knowledge however is predominantly tacit in nature, hence is highly recommended to use rich media to ensure effective KT. As cited by Pedersen et al., (2003), according to Daft/Huber (1987), and Bresman et al. (1999), face-to-face interaction between individuals facilitates transfer of knowledge that is experience-based and permits interactive communication, questioning, flexibility, and adaptation.

E. External conditions surrounding KT

To date, research in KT has received enormous attention especially in investigating the barriers or impediments to effective KT (Ko et al., 2005; Wu et al., 2007; Anna et al., 2009; Paulin & Suneson, 2012). This phenomenon is not surprising since the best strategy to implement effective KT is by identifying and overcoming these impediments. Our study takes slightly different approach in that we are not only determining what the barriers are, but most importantly, we are looking at them from more positive perspectives. We believe that underneath some of the barriers, lays the hidden potential contribution on teams' capability. Therefore, we decide to use "external conditions surrounding" KT instead of barriers. A list of surrounding conditions identified from the literature was explicitly investigated through question 15 to 31. The following Table 6 summarizes the findings for surrounding conditions of KT.

Table 6. Results for External Conditions Surrounding KT

External Conditions	Frequency	Percentage (%)
Physical distance	28	93.3
Functional, experience, and capability differences	23	76.7
Lacking of time	20	66.7
Lacking of trust	18	60.0
Reluctance to share knowledge	13	43.3
Lacking of motivation	7	23.3
Low awareness of the value and benefit of possessed knowledge to others	5	16.7

As predicted, physical distance was the most frequently chosen by the participants as an external condition surrounding KT. This result is in agreement with Gregory et al. (2009) and Anna et al. (2009) who highlight the physical distance as one of the main impediments for effective KT. The fact that two interdependent teams working distantly from one another has definitely reducing the ease for KT. The problem with KT becomes even more acute as more and more issues arose, particularly when the chances for direct face-to-face meeting or social communication, becomes less and less impractical. The fact that software architecture development is a knowledge integration activity, to bridge the physical gap is very important. This explains the previous findings of mediums used for KT, in which various types of communication technologies have been employed to cater the communication problems between the non-collocated teams.

The findings are continued by the selection of functional, experience and capability differences as second most frequently chosen external conditions surrounding KT. Software architecture development witnesses the integration of team members from diverse backgrounds, experiences, and capabilities. In addition, being assigned with different roles and functions has consequently increased the gap between teams. Sarker (2003), in her study found that difference in individual capabilities undermines KT. Reige (2005) also mentions the difference in experience in his study regarding barriers in sharing of knowledge.

The numbers are closely entailed by lacking of time (Roux et al. 2006; Reige, 2005; Ramirez, 2007) as one of the external conditions surrounding KT. A typical nature of software project teams (including software architecture development) does not only confined into achieving specified purpose but also to work within constraints of time. Time restrictions have become the possible reason that drives the teams to hoard their knowledge rather than transfer and share with others. Participants also highlighted the lack of time to engage in KT as a result for being too occupied with the assigned task and reaching the dateline. This comment is consistent with Michailova and Husted (2003), in which according to them, people naturally focus on those tasks that are more beneficial to them. There was one participant who also commented that due to physical distance, they rarely have the time to identify colleagues in need of specific knowledge.

By far, lacking of trust has been nominated by the literature as one of the most common impediments to effective KT (Naftanaila, 2010; Falconer, 2006; Lucas, 2006; Reige, 2005; Hildreth & Kimble, 2004). According to findings in Reige (2005), there are two terms concerning this issue. Firstly, there is a lack of trust in people because they may misuse knowledge or take unjust credit for it and secondly there is a lack of trust in accuracy and credibility of knowledge due to the source, which the latter was studied by Sarker (2002), in her research that investigate KT among information system development (ISD) team members. Naftanaila (2010) asserts that most people are unlikely to share their knowledge and experience without a feeling of trust. This is particularly true when according to some participants, lack of trust is mainly due to lack of social communication between teams, since they are not physically collocated. Social communication often realized through informal networks, which is very limited considering the nature of non-collocated teams. Additionally, "…the nature of inter community social relation…where people have limited sense of shared identity, makes the existence of trust less likely…" (Hildreth & Kimble, 2004)

Reluctance to share knowledge can be possibly caused by the specialized nature of the knowledge both analyst and software architect teams possessed. The specialist nature of their knowledge, combined with the extensive lack of interaction which had been typical, meant that they had very poor understanding of how other functions worked, or what their constraints or requirements were (Hildreth & Kimble, 2004). When asked further about the extent of their agreement concerning this as a reason why there is

a reluctance to share knowledge with others, there were seemed to be no deniable. However, there were few participants who added personal gain and power (job security) as the causes to become reluctant to share knowledge. This finding is in line with Paghaleh et al. (2011). Another finding perceived from the participants concerning the cause for this reluctance is the inability to absorb new knowledge due to incompetence or limitation in their existing stock of knowledge:

"Sometimes, we feel hesitant to share because we are not so sure we can correctly convey to others what we really want to tell them …it is better to keep that to ourselves than giving them the wrong ideas"

Another external condition surrounding KT during software architecture development as perceived by the participants is lack of motivation. There is an indication that it is the primary trigger for KT (Ajmal & Koskinen, 2008; Frey & Osterloh, 2000;). Many studies have been conducted to investigate the extent of effect the lack of motivation has, upon KT (Mclaughlin et al., 2008; Disterer, 2001; Frey & Osterloh, 2000). Lack of motivation, particularly extrinsic motivation has been raised by many as closely related with managerial or organizational issues. This type of motivation is about expected organizational rewards and reciprocal benefits. On the other hand, intrinsic motivation refers to knowledge self-efficacy and enjoyment in helping others and is very important to help perform complex or creative tasks such as developing architecture. In neither ways, both team leader and project manager plays a significant role in cultivating the sense of motivation among team members. In order to fulfill their tasks during software architecture development, KT between teams should be of importance despite of physical distance. An observation reported by one participant regarding this is that KT has always been seen as laborious especially in terms of time and effort. The tendency to fully concentrate in one's work in order to catch the dateline explains why KT is seen in such a way. It is important to note, as is mentioned by Milne (2007), that individuals are often motivated to keep their tacit knowledge for themselves rather than share it. In software architecture development, both analyst and software architect teams need to be able to exploit these tacit knowledge.

The participants also chose low awareness of the value and benefit as one of the external conditions surrounding KT, during software architecture development. One probable reason that drives this issue is that they do not believe these benefits from transferring knowledge. Even worst, they did not actually experience KT although they make claim that they have. As displayed in typical scenario of general software development teams, they often create island of knowledge due to low awareness that the knowledge possessed by the other teams is valuable and useful, which can help accelerate the completion of their tasks. Parallel to this, the intention to transfer knowledge is refrained by the thought that they already possessed a certain level of knowledge, and thus KT is not much in need. When asked their opinion regarding this, the participants were unanimously agreed to have been in such state of condition. A few added by stressing their uncertainty of the presence of KT, due to lack of understanding of the process involved.

IV CONCLUSION

We believe our effort fills in the gap due to lack of understanding and prescription of KT particularly in software architecture development, which consists of analyst and software architect teams that are non-collocated. Future research directions including examine KT in more detail from other different phases in software development life cycles (SDLC); development, testing and maintenance. This strategy allows for a comprehensive view in regards to KT event during software development projects. In order to obtain more concrete lens of KT in software architecture, other roles apart from the analysts and software architects, but are indirectly involved in developing it (including project manager and project leader) seemed to be a fruitful idea of interest to study.

REFERENCES

Ajmal, M.M. and Koskinen, K.U. (2008), ''Knowledge transfer in project-based organizations: an organizational culture perspective'', *Project Management Journal,* 39(1), 7-15.

Andreas, H. P. (2002). A comparison of face-to-face and virtual software development teams. *Team Performance Management: An International Journal* 8(1/2), 39-48.

Anna, W., Bambang, T., Glen, M. D., Chen, L. (2009). Barriers to effective knowledge transfer in project-based organisations. In McCaffer, Ron (Ed.) *Proceedings of the 2009 International Conference on Global Innovation in Construction Proceedings*, Loughborough University UK, Holywell Park, Loughborough University, 220-230.

Bass, L., Clements, P., Kazman, R., Klien, M. (2008). Models for Evalauting and Improving Architecture Competence. *Technical Report.* Software Engineering Institude. Carnegie Melon.

Berlo, D.K (1960) *The Process of Communication: an introduction to the theory and practice*. New York. Holt, Rinehart and Winston.

Boloix, G. & Robillard, P. N. (1995). A Software System Evaluation Framework. *IEEE*, 17-26.

Conroy, G. Soltan, H. (1998). ConSERV, a project specific risk management concept. *International Journal of Project Management*, 16(6), 353-366.

Correa, C. M. (1996). Strategies for software exports from developing countries. *World Development*, 24(1), 171-182.

Disterer, G. (2001). Individual and Social Barriers to Knowledge Transfer. *Conference Proceedings 34th Annual Hawaii International Conference on System Sciences*, Los Alamitos, CA:IEEE Press.

Falconer, L. (2006). Organizational learning, tacit information, and e-learning: a review. *The Learning Organization, 13*(2), 140-151.

Faraj, S., Sproull, L., (2000). Coordinating Expertise in Software Development Teams. *Management Science*, 46(12), 1554-1568.

Gregory, R., Beck, R. and Prifling, M. 2009. _Breaching the knowledge transfer blockade in it offshore outsourcing projects: A case from the financial services industry‘. *Proceedings of the 42nd Hawaii International Conference on System Sciences.* Wikoloa, Big Island, Hawaii

Hansen, M. T. (2002). Knowledge Networks: Explaining Effective Knowledge Sharing in Multiunit Companies. *Organization Science*, 13(3), 232-248.

Harandi, M. T. (1988). Building a Knowledge-Based Software Development Environment. *IEEE Journal on Selected Areas in Communications*, 6(5), 862-868.

Hildreth, Paul; Kimble, Chris (2004). *Knowledge Networks: Innovation through Communities of Practice.* IGI Global.

Jablin, F. M., Putnam, L. L. (2001). *The New Handbook of Organizational Communication: Advances in Theory, Research, and Methods*. Thousand Oaks, CA: Sage Publications.

Joshi, K. D., Sarker, S., Sarker, S. (2007). Knowledge transfer within information systems development teams: Examining the role of knowledge source attributes. *Decision Support Systems,* 43(2), 322-335.

Ko , A. J., DeLine, R., Venolia, G. (2007). Information needs in collocated software development teams. International Conference on Software Engineering (ICSE), 344–353.

Ko, D. G., Kirsch, L. J., & King, W. R. (2005). Antecedents of Knowledge Transfer From Consultants to Clients in Enterprise System Implementations. *MIS Quarterly,* 29(1), 59-85.

Kruchten, P. (2011). Software Architecture for the Business Analyst. Tutorial 3 in the *9th Working IEEE/IFIP Conference on Software Architecture*. Boulder, Colorado, USA.

LaToza, T. D., Venolia G., Deline, R. (2006) Maintaining mental models: A study of developer work habits. *Proceedings of ICSE'06 Shanghai*, 492–501.

Lucas, L.M. (2006). The role of culture on knowledge transfer: the case of the multinational corporation. *The Learning Organization, 13*(3), 257-275.

McLaughlin, S., Paton, R. A., Macbeth, D. K. (2008). Barrier impact on organizational learning within complex organizations. Journal of Knowledge Management 12(2), 107-123.

Michailova, S. and Husted, K. (2003). Knowledge sharing in Russian companies with western participation. *Management International*, 6(2), 19-28.

Milne, P. (2007). Motivation, incentives and organisational culture. *Journal of Knowledge Management,* 11, 28-38.

Naftanaila, I. (2010). Factors affecting Knowledge Transfer in Project Environment. *Review of International Comparative Management,* 11(5), 834.

Osterloh, M., Frey, B.S. (2000). Motivation, knowledge transfer, and organizational form. *Organization Science*, 11(5), 38-50.

Paghaleh, M. J., Shafizadeh, E., Mohammadi, M. (2011). Information Technology and its Deficiencies in Sharing Organizational Knowledge. *International Journal of Business and Social Science* 2(8).

Paulin, D and Suneson, K. (2012). Knowledge Transfer, Knowledge Sharing and Knowledge Barriers – Three Blurry Terms in KM. *The Electronic Journal of Knowledge Management* 10(1), 81-91.

Pedersen, T., Petersen, B., Sharma, D. (2003). Knowledge Transfer Performance of Multinational Companies. Special Issue. *Management Internal Review*, 43, 69-90.

Ramesh, B., Tiwana, A. (1999). Supporting collaborative knowledge management in new product development teams. *Decision Support Systems*, 27(2), 213-35.

Ramirez, A. (2007). To Blog or Not to Blog: Understanding and Overcoming the Challenge of Knowledge Sharing, *Journal of Knowledge Management Practice*, 8(1).

Riege, A. (2005). Three-dozen knowledge sharing barriers managers must consider. *Journal of Knowledge Management,* 9(3), 18-35.

Roux, D. J., K. H. Rogers, H. C. Biggs, P. J. Ashton and A. Sergeant. 2006. Bridging the science–management divide: moving from unidirectional knowledge transfer to knowledge interfacing and sharing. *Ecology and Society* 11(1), 4.

Rus, I., Lindvall, M. (2002). Knowledge Management in Software Engineering. *IEEE Software*, 19(3), 40-59.

Sarker, S., Sarker, S., Nicholson, D., & Joshi, K. D. (2003). Knowledge Transfer in Virtual Information Systems Development Teams: An Empirical Examination of Key Enablers. *Proceedings of the Hawaii International Conference on System Sciences (HICSS-36)*, Big Island, Hawaii.

Sawyer, S. (2001) "Effects of Conflict on Packaged Software Development Team Performance,"*Information Systems Journal, 11*(2) 155-178.

Szulanski, G. (2000). The process of knowledge transfer: A diachronic analysis of stickiness. *Organizational Behavior and Human Decision Processes*, 82(1), 9-27.

Tiwana, A. (2004). An empirical study of the effect of knowledge integration on software development performance. *Information & Software Technology* 46(13), 899-906.

Walz, D., Elam, J., and Curtis, B. (1993). Inside a Software Design Team: Knowledge, Sharing, and Integration. *Communications of the ACM* 36(10), 63-77.

Wei’e, W. (2011). Analysis of knowledge transfer process and model building. *International Conference on E-Business and E-Government. IEEE,* 1, 1-478.

Wu, W. L., Hsu, B. F., Yeh, R-S. (2007). Fostering the determinants of knowledge transfer: a team-level analysis. *Journal of Information Science*, 33(3) 326–339.