Knowledge Sharing for Common Understanding of Technical Specifications Through Artifactual Culture

Mansooreh Zahedi[1], Muhammad Ali Babar[12]
CREST – The Centre for Research on Engineering Software Technologies
[1]IT University of Copenhagen Denmark, [2]University of Adelaide, Australia
mzah@itu.dk, ali.babar@adelaide.edu.au

ABSTRACT
Context: Software engineering is a knowledge intensive activity that is supported by documenting and sharing the required knowledge through a wide variety of artifacts. Global Software Development (GSD) teams heavily rely on artifacts as a vital means of knowledge sharing. However, there is little empirical knowledge about the key reasons and practices of using artifacts in GSD for knowledge sharing to support common understanding of technical specifications. O**bjective:** This study aims at empirically studying the key motivators, practices, and drawbacks of artifact-based knowledge sharing for achieving common understanding of technical specifications in the context of GSD. **Method:** We conducted an exploratory case study in an organization that was involved in several GSD projects. **Results:** Our findings revealed the key challenges that necessitated the use of artifacts for sharing technical specification knowledge. We also present the practices that make up the artifact-based knowledge sharing system in the studied case. Finally, we shed some light on the caveats of knowledge sharing practices adopted by the studied company. The findings can provide useful insights into the artifact-based knowledge sharing practices and how it can be complemented by having certain level of social ties among distributed team members, even through asynchronous means.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management.

General Terms
Human Factors, Management, Process Improvement.

Keywords
GSD, Knowledge Management, Case Study.

1. INTRODUCTION AND MOTIVATIONS
Software Engineering (SE) is usually supported by a wide variety of artifacts that document different types of knowledge pertaining to the software systems being developed or maintained. The artifacts of software documentation are expected to serve many purposes such as aiding development and maintenance tasks [1]. One of the well known benefits of these artifacts is to support a common understanding about different aspects of a software intensive system including its architecture, detailed design, and functions [1]. Whilst there is a wide recognition of having appropriate documentation of a software intensive system, there is usually not a favorable attitude towards documentation based knowledge sharing as documents usually end up being incomplete, outdated, or inconsistent [1, 2]. Growing popularity of agile practices also promote the tendency of focusing on producing executable artifacts (i.e., mainly source code) rather than the artifacts that documents the knowledge about the decisions on requirements, architectural decisions, and detailed design. Research shows that the perception of software development professionals about the potential usefulness of project artifacts varies; for example, comments on the code, data models, requirement specifications, and architectural views are ranked differently as being important artifacts [1]. All of these artifacts can be considered reservoirs of important knowledge about different aspects of the same software systems. Such knowledge needs to be shared for effective and efficient development and evolution of a software system. This study is focused on knowledge sharing for achieving a shared understanding of technical specifications; the technical specifications can consist of requirements, architecture design, and detailed design as defined in [3].

It can be argued that the importance of project artifacts is tightly associated with the context of a software development project and the relevant organizational culture. For instance, the role of project artifacts in GSD projects can go beyond the well-known benefits (e.g., maintenance aid) as compared with the collocated software development projects. It is widely recognized that different distance dimensions (e.g., geographical, temporal, socio-cultural) of GSD inhibit frequent and effective interactions among distributed team members [4]. Unlike collocated software development teams, GSD team suffers from lack of face-to-face interactions that limit their ability to gain a common understanding of software requirements and design decisions through informal and social means of knowledge sharing. Several studies show that achieving a shared understanding of requirements within GSD teams is quite challenging because of inadequate communication [5, 6], cultural differences [5, 6], temporal distance [5], and lack of knowledge management [5].

Team building and maintaining open communication links among distributed stakeholders are raised as commonly used practices to promote knowledge sharing [6-8] for gaining shared understanding [6, 8] of different aspect of a system to be developed. Nevertheless, many organizational and personal attempts to build and evolve social ties for promoting knowledge sharing through frequent and ad-hoc communication may fail for several known (e.g., temporal and cultural distances, lack of trust, and language problems) and unknown reasons. Even in the presence of appropriate social structures, software development project artifacts are considered a rich source of knowledge and many organizations make heavy use of project artifacts for sharing knowledge to achieve a shared understanding of technical specification knowledge within a project team. The tendency of using artifact-based means of knowledge in GSD is much higher

Ease 2014, May 17–18, 2014, London, UK

than normally realized. Despite a pervasive use of software documentation artifact in GSD organizations, there is little empirical knowledge about the main motivators of using artifacts for knowledge sharing and how an artifact-based knowledge sharing system can be set up and operated. It is also equally important to empirically discover the potential drawbacks of relying on artifacts for knowledge sharing. In this paper, we present an exploratory case study aimed at understanding an organizational system of artifact-based knowledge sharing by empirically studying one of its GSD team who had worked on three inter-related projects over the period of almost 18 months. We assert that this is one of the few attempts at empirically exploring and understanding the key reasons and practices of achieving a shared understanding of technical specifications through artifact-based knowledge sharing. Our study, therefore, aimed to explore following key research questions:

RQ1: What are the needs and motivators to use artifact-based approach by GSD teams for sharing technical specifications knowledge?

RQ2: How artifact-based approach is implemented in GSD teams and how does it support shared understanding of technical specifications knowledge?

RQ3: What are the drawbacks of applying artifact-based knowledge sharing approach in GSD?

2. BACKGROUND

In this section, we present related work on main constructs of our study including Knowledge Sharing in GSD, shared understanding, and software documentation.

2.1 Knowledge Sharing in GSD

Knowledge management (KM) refers to the process of creation, capture, transform, deploy and applying knowledge [9, 10]. In context of SE, different KM activities refers to the process of acquiring SE knowledge from different sources and analyzing, capturing and sharing knowledge with different stakeholders [10]. Knowledge sharing is at the heart of KM process that can be assumed as part of KM life cycle from creation to transformation. Nonaka states that knowledge is created during interactions between individuals and converted between tacit and explicit [11]. Hansen et al. [12] distinguishes between two KM strategies: codification in which knowledge is codified and stored in databases to share with others, and personalization in which knowledge is closely tied with individuals and shared through one to one interaction.

In the context of GSD, lack of face-to-face communication and informal chats caused by geographical dispersion can make it difficult to have a smooth flow of information. The process of capturing and sharing knowledge generated at different sites in a GSD project can be quite difficult because of the involvement of multi-culture, multi-stakeholders, multi-languages and multi-processes. Manteli et al. [13] discuss how business strategy, relationships between sites, team structure and work distribution approach could influence creation and transformation of software engineering knowledge among distributed members. The study showed organizational policies to share filtered information with remote sites increases the need of clarifications. Moreover, hierarchical structures, variety of role descriptions and unbalanced team sizes are observed as issues that could introduce notion of "sticky knowledge" to the locations where the majority of competences are located [13]. Boden et al. [14] argue that national, organizational, and professional cultures can impact the process of knowledge creation and sharing among distributed team members. Their study showed cultural and social issues could influence the way knowledge is exchanged such as conducting weekly meetings and sharing minutes versus daily Skype status sessions.

2.2 Shared Understanding

Shared understanding refers to organizing relevant knowledge in a team [8]. This concept has been referred to mutual knowledge in some references [15] as knowledge that communicating parties share and that each party knows that they both possess. Hinds and Weisband [8] describe shared understanding in collaborative work may have different dimensions such as understanding ultimate goal, tasks, anticipated interaction within the team and characteristics of team members. It is evident that a group of individuals with shared identity [8] could develop mutual knowledge and resolve task and personal conflicts [16]. Furthermore, having open communication and smooth knowledge sharing significantly help team members to participate in flow of information and gain better understanding on overall goal, tasks and contribution of other members[8]. Individuals with developed trust and rapport could benefit from ease of informal communication to share knowledge and therefore experience successful collaboration [7, 17]. The studies show that team members who developed common understanding could achieve better performance and personal satisfaction [8, 15]. Nevertheless, shared understanding of distributed team members is threaten by geographical distance, lack of shared context and reliance on communication media [8, 16].

2.3 Software Documentation

Software documentation is an essential part of software development as documents are expected to help organize knowledge of software system. That is why the quality and comprehensiveness of artifacts developed in early phases of a project (e.g., requirement and design) can influence the quality of artifacts in later stages (e.g., implementation) [1]. Documents in organizations could also be viewed as links between people [18] to enable sharing and reusing knowledge possessed by individuals [18]. Documents also help alleviate "*private information*" problems in large organizations [19]. In GSD, documents help standardize practices and develop shared meanings of metaphors, objects and descriptions – as a shared culture [20]. Therefore, there is an increased realization of importance of maintaining project artifacts to support coordination [21] and common understanding of essential knowledge (e.g. requirements) [5].

3. RESEARCH METHOD

Case study is an appropriate method to explore a phenomenon in its real-life context, particularly when the border between the phenomenon and its context is blurred [22]. We investigated our research objective through an exploratory holistic case study [22] of a GSD team.

3.1 Organizational Context

We studied a software development team distributed between Denmark and Philippine who have been initially in client-vendor engagement and recently got merged. The studied project consisted of almost 10 resources involved at each site. The Danish company owns well-established process model in which all the roles, responsibilities, iterations and deliverables are defined. Organizational process considers peer roles at each site, which means roles such as Requirement Manager (RM), Project Manager (PM), Lead Developer (LD) exist at both sites. However,

in practice these roles can be either shared or be merged in one individual. In studied project role of RM at DK is shared between User experience engineer (UX) and LD. Organizational process proposes peer-to-peer communication pattern. Peers get introduced to each other during kick-off and briefed about their responsibilities regarding project deliverables. They are supposed to maintain coordination among their teams and conduct follow-up discussions during project. This project consists of two development tracks, PH-Track and DK-Track. There are two leaders at onshore; each takes care of one track and coordinate inter-track activities among themselves.

3.2 Data Collection and Analysis

Our data collection method included semi-structured interviews and documentation analysis of the projects that the interviewees had carried out in one team. We received access to several documents from the two projects. We carried out 15 semi-structured interviews in two phases. During the first phase, we carried out 12 interviews – 6 interviewees from each site (i.e., Demark and Philippe). For the second phase, we carried out 3 more interviews after 4 months of the first phase. We did not carried out further interviews in the second phase because of data saturation. We interviewed different roles at each site that include project managers, leaders, requirement managers and couple of developers. The second phase interviews were conducted with the technical lead, project manager, and user experienced engineering located onshore. We scoped our study to early phases of the project life cycle; from requirements elicitation until the start of development, as these phases are known to be the most knowledge intensive phases. Our interview questions mainly uncovered how different roles get engaged in capturing, analyzing, sharing and applying knowledge over these phases and perceived challenges. We also asked about communication and social interactions for understanding informal knowledge sharing channels. We audio recorded all the interviews that resulted in approximately 400 pages of verbatim transcription.

We analyzed the data using qualitative data analysis approach called thematic analysis [23]. Having reviewed the transcriptions and getting familiar with the data, we performed coding using Nvivo10. We performed coding by extracting data fragments indicating the process, through which the requirements are captured, analyzed, documented, shared, clarified and the design solutions are proposed. We also coded scattered quotations referring to the needs and motivations behind certain practices and associated challenges. Gradually collating and refining the emergent codes resulted in themes for identifying the key project documents, knowledge sharing practices to mitigate misunderstandings, needs of applied practices, and the associated challenges. We also analyzed 1-2 key project documents (i.e., product descriptions, design specifications, organizational process model) to verify the interviewees' statements and to deepen our understandings of their structure and contents.

4. FINDINGS

We present our findings to answer the research questions for this study. The findings identify the reasons of sharing technical specifications through artifacts. The findings also reveal how the artifact-based knowledge sharing system was adopted to gain shared understanding of technical specifications in the studied case. The drawbacks of the artifact-based knowledge sharing systems are reported along with some recommendations.

4.1 Why Artifact-Based Knowledge Sharing?

In this section, we report the reasons for which the studied organization adopted the use of artifacts for sharing knowledge for gaining a common understanding of technical specifications. Despite the organization had duplicated all the key roles on both sides, it was felt that they needed to emphasis the role of different project artifacts (e.g., product description) for communicating technical specifications knowledge. The artifact-based knowledge sharing had its roots in the lack of open communication, complexity of domain knowledge, technical knowledge imbalance, and misgivings based on previous experiences.

4.1.1 Reluctance to Openly Communicate

Our analysis revealed that the studied organization had tried different strategies to promote open communication and frequent personal interactions for building social ties and knowledge sharing after acquiring its vendor company. The tried strategies included organizing briefs on cultural differences and how to leverage them, encouraging video-conferencing and instant messaging, and sending some staff to visit the development site in Philippine. There was also an attempt to simulate informal chats at coffee machine by having a live camera and screens installed to cover the coffee machines and water coolers at both sites. However, this solution did not help to trigger natural social interactions. There was also an internal social networking site, called face it, for sharing profiles of the distributed colleagues and enabling one-to-one communication. Whilst, these strategies showed some improvement in promoting cross-site interactions, there was no progress on frequent, ad-hoc and open communication at the level of individuals; and it was considered important for sharing knowledge for gaining a common understanding of the different aspects of the technical specifications. We found both sides reluctant to have frequent and open communication for sharing domain and technical knowledge; a general lack of confidence and shyness of the offshore developers appeared to have contributed to this situation more than their Danish counterparts. The offshore developers preferred talking to onshore through their managers, which was attributed to their hierarchical culture. *"I don't think offshore developers tell their challenges to Danish project manager because of culture. During my stay at offshore I learnt that it was not as easy to talk to developers as it was to talk to their project managers"PM1,*

Furthermore, lack of confidence in communication (stated by an offshore developer as *'the expertise on how I should express my concern with other developers'-Dev1)* and different working habits to directly confronting clients also caused uneasy feeling of offshore developers in direct interaction with Danish colleagues. *"When we ask offshore developers do a presentation, obviously they are quite nervous because it's not something that they normally do. A Danish developer used to confront a client and doing analysis, that's a consultant kind of profile but developers at offshore are not used to that kind of work. They normally talk to their leader who describes the tasks."PM2*

At the Danish side, the reluctance to frequent communication was caused by the new experiences of engaging in a new development arrangement; moreover, there were also difficulties to interact over the available communication media. *"I think it was a big change in Denmark, because we have a history of doing all the development in-house. All of a sudden we have to collaborate with another partner. It requires that we have a changed mindset here in Denmark…it's very difficult for Danish project manager to have a day-to-day contact with offshore developers. It's difficult*

to set up a videoconferencing and get good understanding of their progress and their issues" PM1

In absence of frequent and open communication between distributed sites and high chance of misunderstandings, artifacts (e.g. design specification) are used as more reliable means to explicitly exchange knowledge on different aspects of system and avoid miscommunication.

4.1.2 Complexity of Business Domain Knowledge

The studied company develops software systems predominantly for the Danish public sector in Denmark. The systems developed are expected to support hundreds of major and minor rules that characterize the social support system in that country. Hence, the domain knowledge is rather complex to easily understand unless being in that country. For example, the studied team developed software to support food inspections in places such as restaurants, slaughterhouses, and farmlands. The onshore Requirements Manager (RM) and User Experience engineer (UX) would acquire the domain and requirements knowledge through extensive interactions with customers in several workshops. Whilst RM shares most of the domain knowledge with the collocated developers through face-to-face interactions whenever required, it has been difficult to share the domain knowledge through conference calls (video or audio) or emails. For the onshore team members, it appears to be not a cost effective to organize domain knowledge sharing sessions. *"If I spend time upfront transferring domain knowledge, I probably have to do it on recurring basis as it would be forgotten. I'm sure we would benefit but not sure it would be greater than cost"PM2*

That is why the RM and UX found it relatively inexpensively and easy to explicitly document as much domain knowledge as they feel necessary for sharing with offshore team members. However, it is quite challenging to make the domain knowledge explicit for the offshore team members. It is not easy to decide the types of context and domain knowledge that needs to be shared. *"We have a lot of domain knowledge in Denmark about how we expect something to be done. For example, we had to develop a draft analysis so user can save a draft…when client told us we pretty soon understood, but when we told to Philippines they had a completely different understanding of how a draft would work. So that requires us to explain a lot of this implicit knowledge." PM3*

4.1.3 Technical Knowledge Imbalance

There is a significant imbalance in the technical knowledge of onshore and offshore; it makes the knowledge sharing through discussions and interaction difficult as the Danish developers have more knowledge of the technologies being used because of their long working histories compared with their relatively young and inexperienced counterparts. The company has classified software development staff offshore into seniors and juniors unlike onshore where most of the staff has titles (e.g., developer and technical lead) but no classification. Hence, the technical lead at offshore needs to explicitly describe what to do and what not to do with the technologies being used. One of offshore developers acknowledge their relatively less knowledge and experience. *" N-tier architecture of the system is new to me, it is very complicated, between five layers I have to implement different business rules that are given in the documents sent from Denmark…"Dev2*

The newness of the technical solutions used, programming language, and quality standards for coding introduce high learning curve for the offshore team whose only sources of asking technical questions are the offshore technical lead and the detailed documentation prepared by the onshore technical lead, as they are too shy to directly communicate with the onshore team members. A realization of the technical knowledge gap has convinced the onshore management that the best solution to provide the required technical knowledge and guidelines is through project artifacts such as design specification and development handbook. *"Database programming is one of the areas that I felt there were no improvement. They do it too complicated and sometimes wrong. So we spent extra effort to put some guidelines based on our experiences in handbook"LD2*

4.1.4 Misgivings Based on Past Experiences

Previous attempts of relying upon personal interactions between different counters to share knowledge and contextual details had created miscommunication and misunderstanding with regards to the technical specifications to be implemented and the required expertise. That situation cost the company quite heavily for fixing the errors and bringing most of the development to onshore for the complex parts of a large project. The main reason for that situation was a relatively poor provision of requirements by onshore and too much reliance on the offshore team for specification analysis and design activities based on the contextual and domain knowledge mostly shared through interactions. One of the project managers explained this experience as: *" We made a decision that entire system should be developed from PH. When we got into details with the requirements, we started to move the things back to DK based on complexity and if there were special knowledge needed in relation to Danish public sector."*

Apart from the misgivings from the experiences just mentioned, there was also an impression of lack of competency in writing high quality code. And it was decided to extensively use artifacts (e.g., development guide) to provide illustrated examples of high quality code and expected quality standards for designing detailed algorithms as such knowledge appeared to be difficult to share through other ways. *"The quality of the code they develop is sometimes very poor."Dev4, "They do quite complex procedures and design of database. That is why we need to write detailed examples for them" Dev4, LD2*

4.2 Artifacts for Knowledge Sharing

With regards to the questions like what kinds of artifacts and their content related to technical specification knowledge, our analysis focused on the early lifecycle development processes of the organization. Since most of the technical specification knowledge is generated during requirements elicitation and design solution activities that are predominantly performed in Denmark, a majority of the technical specification knowledge creation and capturing takes place in Denmark through extensive and long interactions with customers. Then the respective project staff (i.e., RM, XU, and PM) creates different artifacts to share that knowledge with offshore counterparts.

Onshore UX and technical leader, both sharing the role of RM, organize requirement acquisition workshops with customers. Each iteration starts with such workshops over several days but limited to 15-20 hours in total. The workshops aim to discuss business domain and identify customers' needs. The technical leader makes extensive notes on the technical requirements during the workshops. And UX captures business processes and contextual information by making sketches of the workflows and the required user interfaces. These drawings are used to facilitate discussions during workshop sessions and incrementally refined with customers' feedback in live sessions and offline communication. The finalized drawings are documented as one of the key project artifacts called Information Architecture (IA). The

IA mainly captures the domain knowledge and key decisions about the user interface of the system consisting of sketches and snapshots of drafted interfaces and annotated with real inputs (e.g., select lists) and comments. As both the roles described: *"The workshops are all around the user interface and how they should work with the system. We used drawings based on customers' use cases and made drawings as start point for dialogue and then we just commented the drawings and documented in wide frames" UX1, LD1.*

Table 1 - Overview of Key Project Artifacts and associated knowledge about technical specifications.

Key Artifacts	Brief Description
Information Architecture (IA)	Sketches & snapshots of information flow with related comments. Captures Business Process and User Interface knowledge.
Product Description (PD)	Describes functional requirements and business logics of a product. Written separately for each development track.
Design Specification (DS)	Describes detailed design solution (e.g., data model structure, reports formats, mapping the fields) to develop a product. Written separately for each development track.
Development Handbook	Describes architectural solution, quality conformance rules (e.g., naming conventions, use of database objects), tools configuration (i.e., version control & shared repository).

Onshore technical lead and architects analyze the workshops' key discussion points and propose technical solutions. Through an iterative process, architecture is designed, reviewed and approved by customer. During this phase, the collected knowledge of requirements is broken down into workable units that are called *product*. Products are considered reservoirs of important technical specification knowledge required for project planning, distribution of work between the sites, tracing the bugs, changes, and release planning. Each product details lead to decisions about work distribution with minimum interdependencies of tasks between development tracks to be developed onshore and offshore. After the work distribution decisions made and captured, the onshore technical leaders document them in an artifact called Product Description (PD), which is one of the key project artifacts. Preparing separate PDs for each of the development tracks is considered necessary for making elaborated documents for offshore team. To ensure the correct of the technical specification knowledge captured in PD, customers are asked to review them.*" PD includes business requirements of the system that customer can demand."LD1;"it contains functional requirements and business logics" PM1, PM2;"They are in English and translated by customer for review and approval" LD1, PM1, PM2; "there is balance between keeping it at business level description, but also detailed enough so nothing is missed, for example we even say here we have pop-up window" LD1," We keep PDs on SharePoint and correspond to that define tasks on Team Foundation Server" LD1*

Offshore development team gets engaged after receiving IA and PDs from onshore as the key sources of technical specifications. They analyze these documents and share their understandings of the requirements with the onshore team, mainly RM and UX, in solution walk-through sessions via videoconferences. All the participants go through IA and PD and share clarifications about the business domain and functional requirements. After the walk-through sessions, a detailed design solution is prepared in Denmark. During this activity, the onshore technical lead of the PH-Track produces another important artifact called design specification (DS). The DS captures technical details such as elaboration on data model (i.e. database design, tables, fields), format of reports and mapping of fields to data base columns.

DS is also an instrument for sharing architectural knowledge by detailing the implementation constraints to comply with architectural decisions. This document is mainly prepared for the offshore team as the onshore team closely interacts with their technical lead in preparing such a document at abstract level. .*"DS is technical design that is very much in details, for example, describing database fields and mapping user interface to the fields. It is not reviewed by customer so can be detailed as much as we want" LD1, "It is written in English by leader in Denmark and updated by offshore leader"PM1, LD2, PM2. "It goes with PD to offshore to say how solution should be designed and developed" PM2*

The onshore technical leader also produces a detailed document called development handbook. This artifact describes architectural solution (i.e., called reference architecture) and implementation guidelines to ensure conformance of design and code to the reference architecture, standardization of codes, and appropriate use of tools. The handbook proposes rules for using database objects (e.g., functions), debugging and quality checks, naming conventions, and standards for configuration of shared repositories. Both development tracks are supposed to review the handbook and follow the provided guidelines: *"Project handbook or development handbook is prepared to capture the decisions and expectations about detailed design and coding including reference architecture, how will layers be structured, what technologies will be used, and dos and don'ts of coding" LD1*

Table 1 provides an overview of the key project artifacts that are prepared and used for sharing technical specification knowledge with offshore project teams in the studied company.

4.3 Artifact-Centric System for Knowledge Sharing

Based on our analysis of the use of the artifacts by different actors and interactions among those actors to share technical specification knowledge, we have characterized an artifact-centric knowledge sharing system, shown in Figure 1, for gaining shared understanding of technical specifications among distributed teams. The system consists of a set of actors, artifacts, connections among them, the adopted practices for shared understanding, and the supporting technologies.

4.3.1 Layers and Elements

Actors layer represents the structure of the distributed teams and their interactions. The findings show that due to existing peer-to-peer communication pattern across the sites, the inter-site interaction mainly goes through RM and technical lead of both sites. Offshore developers talk to Denmark through their team lead and RM. In Denmark, UX and technical lead for the PH-track (i.e., both sharing requirement manager role) are the main contact persons to Philippine. Since the tasks between onshore and offshore teams are loosely coupled, there is no connection (i.e., direct communication) between the development tracks at level of

developers. That means the required inter-track coordination takes place between Danish PH-track leader and the development team in Denmark. At each site, there are a lot of interactions among the collocated team members for social and technical discussions; however, those were not the focus of this study. Customers interact with the Danish side only. We should note that figure1 only represents dominant communication structure based on self-reported data and exceptional interactions have been ignored for the sake of clarity.

Artifacts layer provides an overview of the key project artifacts that carry the technical specification knowledge. We consider emails and conversations of distributed team members as lateral artifacts that are exchanged through the communication patterns indicated as two-way arrows in Figure 1. The comments on the code, as lateral artifact, are the only knowledge-sharing channel among distributed developers. Our analysis of the data reveals two types of linkages: actor-artifact and artifact-artifact. The actor-artifact links aim to share explicit knowledge about business process, functional requirements, technical solution, and guidelines. Geographical and temporal distances among team members make these links more critical for shared understanding of the technical specification knowledge. The onshore technical lead of the track developed in PH plays a critical role in this knowledge sharing system as he/she produces and maintains all the key artifacts. The actor-artifact links are also considered communication basis between the actors who do not have direct interactions. For example, our findings show that detailed technical concerns of onshore leader (e.g., database programming rules, naming conventions) are shared with offshore team through development handbook. The development has helped the development teams to gain a shared understanding of the quality conformance rules and standards without directly interacting on this issue. The artifact-artifact links facilitate inter-relation of different pieces of knowledge produced during the process. The noticeable link is keeping the relation between functional requirements (PD) and detailed design specification (DS). These links enable traceability between different pieces of knowledge for offshore development team who is not actively involved in the creation of those artifacts.

Infrastructure layer presents an overview of the technologies that have supported the interactions of the actors and the artifact linkages. The studied team makes heavy use of Microsoft SharePoint as artifacts repository, knowledge sharing activities and other main asynchronous interactions. All the artifacts are organized in shared folders to enable the required linkages of artifact-artifact (i.e., link to local websites) and access of actors to artifacts. Emails have also been used for sharing some of the artifacts and updates on shared repository. The Audio/ Videoconferencing facilities and screen sharing are other technologies that support the artifact-centric knowledge sharing activities between two teams. And productivity tools (e.g., Microsoft Word, PowerPoint, Visio) are used for preparation and maintenance of the artifacts.

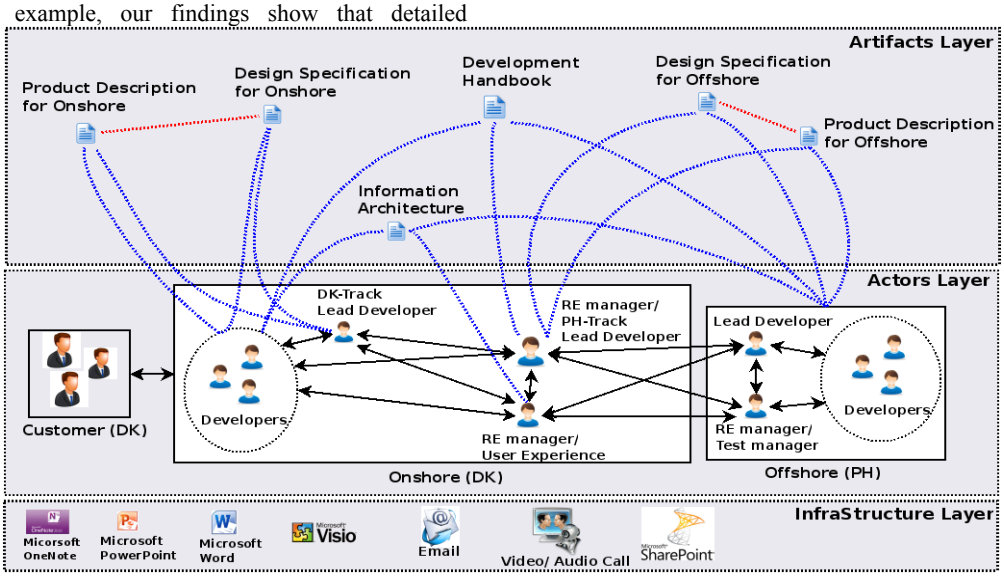

Figure 1 - Artifact-Centric System For Common Understanding

4.3.2 Knowledge Sharing Practices

The artifact-based knowledge sharing system is supported by different practices to minimize the risk of misunderstandings. The noted practices are complexity assessment of the required knowledge, provision of complete artifacts, verifying shared knowledge, and monitoring alignment of artifacts and updates.

Complexity Assessment of the Required Knowledge: The onshore managers attempts to minimize the chances of misunderstandings by reducing the need of complex domain knowledge at offshore. For example, the parts of a system that needs to implement dozens of legal rules and exceptions or the parts that are tightly associated with the existing systems (e.g., integrations and data migration) are not assigned to offshore team: *"If there are a lot of business rules, then there are a lot of exceptions to the flows so I would characterize it as a complex system. But if it's a relatively simple flow with very few exceptions and very few additional business rules then I would say it's low complexity." PM1*

Whilst the requirement-engineering activities are always done by the onshore team, the complexity criteria are applied to the involvement of offshore site into the solution design activities. Most of the architecture and detailed design activities are also performed at onshore in order to minimize the risk of misunderstandings due to the knowledge gap between two sites. This approach appears to help gain a good understanding of the tasks to be allocated to offshore by providing structured skeleton. *"We decided whole database design would be done in DK because we had the domain knowledge so we could make a better model. Also, the overall solution, architecture and design are*

done in DK. So there would be a skeleton of the solution ready and Philippines could say, "Ok, I could see your example" like a hello world " LD2

The complexity assessment is performed before the work distribution. This assessment is done informally by drawing different flow charts from business process scenarios and discussing the required effort for understanding domain or close collaboration for technical inputs from the onshore team for the offshore team. *"We go through products, one by one and decide whether or not this is feasible to do that by offshore. If it is for example integration, data migrations, complex calculations or something that is very dependent on business knowledge we'll do in Denmark." PM1. "All around the user interface was developed at offshore, because it's easier to hand over visual specifications rather than very technical and complex specification" UX1*

This practice also saves time and effort from onshore team leader to prepare detailed artifacts for sharing the required domain and technical knowledge with the offshore team. *"It's not about their skills, if we have a task that is complex and we have to hand it over to someone that is located in another site, we have to use a lot of time describing all the details. But if we develop here, requirement manager can talk directly to developer. They can go to a room, make drawings, so it's the communication and handover activities that become much more time consuming for complex tasks, not lack of skills or lack of quality."PM1*

Providing Comprehensive Artifacts: One of the notable practices to mitigate the risk of potential misunderstandings at offshore is to make knowledge as explicit as possible by providing detailed documents to the offshore team. These documents include business process, elaboration on functional requirements and solution description, detailed design, and development guidelines to represent adoption of this approach. The existing knowledge gap (i.e., business domain and technical) between distributed team members and lack of open communication have made these artifacts as the main source of exchanging knowledge. Our analysis of the artifacts and interviews revealed that the product description artifacts very lengthy and detailed for the offshore team. *"We will adopt more detail in requirements because it's a lot harder for PH to gain business domain knowledge. It's not easy for them to interpret requirements that are in high level and can't ask the customer. They'll ask us. So it's easier to detail requirements to begin with than having to do it all along and also risk of PH developers having misunderstanding." PM2*

As previously reported, the onshore team leads the design of the overall technical solution without any significant involvement of the offshore team. Hence, they realize the important of providing detailed design specifications and guidelines for the offshore team to support the development tasks. The detailed design specifications, therefore, are considered as the main instruments to share essential knowledge of the solution to offshore and decrease the need of ad-hoc communication for seeking clarifications.*" We never detail something as detailed as these [like] specifying the field and what is matter to what, but this is to ensure that they can start developing and they don't have to be on the phone [with us] every 5 minute to ask what about this field? What should that match to?" LD2*

Verifying Shared Knowledge: Both sides attempt to verify the correctness of the shared knowledge and clarify ambiguities. One of the significant practices is organizing *"Reverse Presentation"* during the walk-through sessions. Reverse presentation means having the offshore team to present their understanding of the required functions based on the product specification provided to them. For this activity of knowledge sharing, the offshore team members share their understandings of the provided artifacts (i.e., IA and PD) by giving brief presentation of the proposed implementation plan. The presentation slides usually have UML diagrams (e.g., class, use cases) and pseudo code of database functions to support the discussions. *"We make PowerPoint presentations. We take all the scenarios they have written in the product descriptions. And we prepare how we are going to do implementation and how we understood the requirements that they have written and discuss with them" Dev1*

The reverse presentation preparation implicitly forces the offshore team members to read all the relevant artifacts and reflect upon them. Therefore, the reverse presentation not only brings early feedback to the offshore team, but also increases awareness of the onshore site about the effectiveness of the artifacts used for sharing the knowledge. This practice has successfully helped the distributed teams to build common grounds of requirements and tasks. *"In the start of iteration we sent the product descriptions to Philippines and then had hand-over meeting. In that meeting we wanted PH to present for us their approach and how they were seeing what to implement instead of us presenting to them what they should do because when we turned that around it was clearer for us if the PH understand how to do and what to do. " UX1, "So it was a great success"PM1, PM2, LD1, UX1*

Monitoring Alignment of Artifacts and Updates: The artifact-based knowledge sharing system is supported through extensive monitoring and verification activities to ensure alignment of different artifacts and avoiding misunderstandings. The technical lead and RM are responsible for reviewing different project artifacts and clarifying any mismatches with their peers at the other side. For example, mapping PD against IA (*I analyze the requirements in PD and IA and see if the business rules are complete or any scenario is missing"UX2),* verification of detailed specification against PD and data model (*said by UX2, LD2)* and checking the design artifact against the development handbook (*said by LD2). "I use the detailed specification to check details of all the fields. For example, if the PD says text field should be limited to 10 character, I check if the same thing is said in detailed specification and there are no conflicts" UX2*

The verification efforts through peer roles have helped distributed teams to identify and resolve misunderstandings of artifacts. *"There were cases that PD from business point of view says we have to retrieve all information and copy to this table, but it was not defined in technical specification, so we investigate those cases and clarify with our peers before proceeding."LD2*

4.4 Caveats of Knowledge Sharing Through Artifacts

Like all other ways of sharing knowledge, the studied system of artifact-based knowledge sharing can have drawbacks. We analyzed the gathered data to identify the drawbacks in order to report them as caveats in the following sub-sections

4.4.1 Extra Effort and Cost

One of the significant cons of the studied artifact-based knowledge sharing system is the huge amount of time and effort that are required for preparing and maintaining detailed documents. Whilst it is known that the approach is costly, it is considered the only reliable solution to mitigate misunderstandings and to minimize the cost of bug fixing later. *"The main issue is we have to spend so much time balancing our*

details. If we do too much documentation, then we always have to maintain a lot of documentation during the project, which takes time and slows down everything. But, on the other hand if we don't document enough, there are often misunderstandings." PM3

Moreover, it also burdens the key technical lead with a lot of work for producing and maintaining the artifacts required for sharing knowledge and it can impact his/her participation in analysis and design activities of different iterations. *"Our leader had to do both analysis with customer and concurrently do all detailed specification and handing over to PH at the same time. That brings up some problems with his limited time" LD1, PM2.*

4.4.2 Dependency on Design Knowledge

The artifact-based knowledge sharing and the application of complexity criteria have increased the dependency of offshore team on the sources of technical knowledge at onshore. Lack of domain knowledge and provision of detailed design solutions limits offshore team's autonomy in making design decisions based on their own understanding and knowledge. That is why there is a strong dependency of the tasks assigned to the offshore team onto the inputs provided by the onshore team despite having loosely coupled linking between the tasks assigned to two sites. One of the major issues faced by the two teams is related to design and maintenance of data model. The onshore team takes the ownership of designing and maintaining database. It introduces delay in clarifying questions or getting approval of any required changes in the database model or implementation. "*In this project, the architects who designed the database were from DK team. We're actually using the DB. So in terms of changes we need the approval or at least talk to the architect who designed the DB before we can implement any changes. Whereas, if the project was collocated, we could ask right away whether or not this change could be done, if there's potential impact and so on*" Dev2

The PH development team faces difficulties in properly using the designed database and address the quality concerns because they have different problem-solving approach and lack the rationale of design decisions. This situation impacts their understanding of the data model and causes frustrations. *"They designed the database and we were not part of it. So they just sent us the data module and informed us that these are the tables and relationships but we don't know how the design is optimized. Because, here we usually do performance testing by populating database with lots of data, but we have not been involved in data base design."LD2*

While lack of design knowledge at offshore increased the need of having more detailed design specification, it also increased the dependency of offshore team for clarifying the doubts, reporting mismatches in the documents, or getting approval for changes. This situation therefore, became more challenging in case of unavailability of key people due to sickness or taking long vacations (e.g. 3-5 weeks).

4.4.3 Difficult to Bridge Technical Knowledge Gap

The artifact-based knowledge sharing does not help in bridging technical knowledge gap despite extensive effort from the onshore leaders to provide detailed guidelines for conformance of implementation to architectural decisions and quality standards. Since there was no possibility for the offshore developers directly listen the key points of the guidelines to be followed by the offshore technical lead, those instructions were not followed or understood properly by the developers. It was felt that the root cause was the limited skillsets of the offshore developers to develop the code according to the prescribed rules. It appeared that the guidelines did not motivate the developers to fully understand and follow them. It introduced rework, i.e., performing code cleaning up by the offshore technical lead. *"In the beginning I thought they have not followed handbook at all and no one read it. But it occurred to me, they have developer does the implementation and another developer or senior lead developer will take a clean up and fixing all the violations of the handbook. It's kind of making other people responsible for that and I have to wait maybe a month then they went through all the steps." LD1*

Another barrier to bridge the technical knowledge gap was absence of communication among developers at both sides. Detaching the tracks lead to information silos and inhibited emergence of knowledge sharing channels at the developers' level where most of the knowledge gap existed. *"Communication with Danish developers can be a vital part of my growth in becoming a developer, because they have tons of things on their experiences, and communicating with them regarding our workload or tasks would help me grow and would help us in implementing well, working efficiently with minimal bugs" Dev3*

The use of artifact as the main means of communicating the technical knowledge from the Danish technical lead to the offshore developers decreased the chances of learning about the knowledge limitations and difficulties of the developers. Hence, there was no significant attempt to help them to gain the required knowledge on the company's expenses, a common practice for the Danish side of the developers. This problem was gaining the attention of onshore managers.*" [In future] I'll be probably involving PH more actively. I mean we could have had meetings where I could ask PH to go through the code and tell me how this lives up to the reference architecture and things described in handbook. That will make awareness, […] make them more aware of how they implement the things" LD1*

5. LIMITATONS

The potential impact of construct validity threats [22] was minimized by using different sources of data including semi-structured interviews and project documents. Conducting interviews with different roles from both locations uncovered wide range of perspectives about the project practices and challenges. We also analyzed the related project documents (e.g., sample design specification and organizational process model) to obtain bigger picture of the points raised about the project artifacts (e.g., contents and detailed level).

We tried to ensure reliability [22] by audio recording and verbatim transcribing all interviews. While the findings mainly result from researchers' interpretation of the data [24], we maintained our ongoing internal discussion to verify the findings for decreasing the risk of misunderstandings.

Generalization is one of major barriers to single case studies [22]. Typically critics compare generalization of case study findings with survey research in which samples are generalized to bigger dataset. Yin [22] argues the need of differentiation between analytical (i.e., case study) and statistical (i.e., survey) generalizations. Findings from a single case is dependent on the context in which study is conducted, thus reproducing similar results within another case requires a similar context.

6. DISCUSSION AND CONCLUSION

One of the goals of our research is to inform software development professionals about different means of sharing knowledge for achieving a shared understanding of technical specifications for developing and/or evolving software intensive systems. This case study was motivated by our observations that

knowledge sharing through social interactions and informal communication in GSD may not bring optimal results at best and may cause project failures at worst. That is why many organizations involved in GSD are inclined to share knowledge through artifacts that document technical specification and knowledge about the key decisions made about the technical specification, for example, requirements prioritization decisions. This case study has identified some of the key reasons for adopting artifact-based knowledge sharing for gaining a shared understanding of technical specifications. The studied GSD team adopted artifactual culture of knowledge sharing as per organizational practices despite being working on their third project within the same domain and having known to most of the project staff for almost eighteen months. The Danish team was cognizant of the importance of supplementing their artifact-based knowledge sharing with personal interactions, but they did not want to take any risk as previous attempts had caused many misunderstanding costing the company a lot of money.

Our case study has revealed that the studied GSD team had a history of challenges in sharing business domain and local technical knowledge. Their offshore team members' perception and etiquettes of informal communication were heavily influenced by their cultural norms and history of being a vendor for their parent company. That was why it was not possible to share the required knowledge through frequent communication and interactions as was being done at the onshore (i.e., Danish) site where developers were sitting next to RM, PM, and XU. Hence, the developers could easily approach to any one in the local project team to seek some information or explanation about technical specifications being implemented. Rather most of the technical specifications and the key decisions associated with them were made in developers' presence. This was not the scenario for the offshore team – a common scenario in GSD projects. However, it was important for the whole team that all of members are provided with opportunities to gain a shared understanding of technical specifications because sharing domain knowledge among distributed team members is a crucial factor in productivity, quality, and cost of a project [25]. We also observed that there were several collaborative problems between two sites as a result of thin spread of domain knowledge; a situation that has also been observed in other studies [25].

The experienced and perceived challenges in sharing knowledge through personal connections and communication motivated the studied project team (and the whole organization) to apply different mitigation strategies to avoid the risk of creating misunderstandings about the technical specifications across two sites. They decided to create and share several artifacts for sharing knowledge with the offshore team members. One of the shortcomings of the artifact-based knowledge sharing is that it is quite difficult to detach knowledge from its context for documenting and sharing. We also realize that group collaboration requires sharing task, social, and contextual information. While these types of information are easily exchanged in collocated teams, they are quite difficult to share with distant team members [26]. Hinds and Weisband [8] state that a group of collocated people with similar background (e.g., personalities and interests) and shared experiences (e.g., working together) are more likely to build shared understanding by developing a group identity, easily exchanging the required information, and openly discussing and resolving misunderstandings. However, it is well know that shared understanding within a GSD team is negatively impacted by geographical distance, different contexts, and reliance on communication technologies [8, 26]. These were similar challenge that motivated the studied team to rely on an artifact-based knowledge sharing system.

The studied team adopted a structured and consistent mechanism of documenting and sharing technical specifications knowledge about which all the team members had a shared understanding as each of them knew exactly what information to capture and/or found in which document and how to consult or update each of the documents. Despite having reliance on the artifacts, the studied team members also tried to share knowledge through peer-to-peer communication but at the managerial levels only; the offshore developers could not be convinced to have a direct communication with their offshore PM team. Hence, a clear reluctance on both sides to develop a close rapport caused by hierarchical culture and history of client-vendor relationship decreased the chances of building strong social ties to support shared understanding through sharing knowledge informally [7, 8]. The strategy of applying complexity criteria and getting tracks developed in isolation appear to have negative impact on the development of group identity among the distributed team members; this situation is also a significant inhibitor to knowledge sharing. Hinds and Mortensen [16] discuss that the shared group identity in distributed teams help resolve interpersonal as well as tasks conflicts. They argue that people with similarities tend to partition colleagues as "in the group" or "out of the group". Individuals who are "in the group" are more likely to frequently share information with each other, having a certain level of trust.

Having realized the significant human-centric barrier to promote knowledge sharing for supporting shared understanding of technical specifications, the studied GSD team's organization decided to pay emphasis on the role and importance of artifacts for sharing domain and technical knowledge; they built an effective systems of artifacts that are meticulously prepared, rigorously reviewed, and consistently updated. The key artifacts of this system are Information Architecture (IA), Product Description (PD), Design Specification (DS), and Development Handbook. Our research has shown how the artifacts are created during the initial phases of a project and the types of technical specification knowledge that is captured and shared through each of the key artifact in the studied system. Our findings have also revealed that the studied team has been using the artifact-centric system quite effectively for sharing knowledge and training.

Whilst the studied GSD team appears to be quite satisfied with their use of artifacts for knowledge sharing, they also realize that there are several potential drawbacks of too much dependence on artifactual culture for knowledge sharing. Our study has revealed some of the key drawbacks of such a system. Some of the participants also shared their concerns about not being able to develop social ties and informal communication links for supporting shared understanding. The adopted approach leads to duplication of roles at both sites that means increased project cost.

We can also conclude that filtering shared knowledge and separating development tracks have resulted in information silos with less learning possibilities for offshore development team. This distinction inhibits sharing experiences for developing shared identity and context. Hence, we recommend that the current artifact-centric system be supplemented by promoting spontaneous communication among distributed teams including at developers' levels. Maintaining spontaneous communication among GSD teams could help to develop shared context and identity and result in resolving task and interpersonal conflicts [16]. This mechanism could help distributed team members incrementally grow up in their respective roles and feel personal

satisfaction [7, 15] from group collaboration. Moreover, the distributed teams should get more engaged in collaborative work rather than isolated works to feel motivated for frequently interacting with each other and initiating spontaneous communication. We can also conclude that the adopted mitigation strategies to avoid misunderstandings appear to be effective and can be assessed by other organizations for implementation in their GSD projects. It is also important that the offshore team members are involved earlier in acquiring and specifying contextual and technical knowledge. GSD teams should also emphasize open communication and socialization practices to enable informal knowledge sharing for gaining shared understanding [6, 8].

ACKNOWLEDGEMENTS

We are grateful to the participants of this study. This research has been partially funded by the project "NexGSD" through grant number #10-092313.

7. REFERENCES

[1] G. Garousi, V. Garousi, M. Moussavi, G. Ruhe, and B. Smith, "Evaluating usage and quality of technical software documentation: an empirical study." pp. 24-35.

[2] C. Treude, and M. A. Storey, "Effective communication of software development knowledge through community portals." pp. 91-101.

[3] I. Sommerville, *Software engineering*: Addison-Wesley, 2007.

[4] M. Ali Babar, and C. Lescher, "Global software engineering: Identifying challenges is important and providing solutions is even better," *Information and Software Technology,* vol. 56, no. 1, pp. 1-5, 2014.

[5] D. E. Damian, and D. Zowghi, "RE challenges in multi-site software development organizations," *Requirements Engineering,* vol. 8, no. Copyright 2004, IEE, pp. 149-60, 2003.

[6] D. Damian, "Stakeholders in global requirements engineering: Lessons learned from practice," *Software, IEEE,* vol. 24, no. 2, pp. 21-27, 2007.

[7] J. Kotlarsky, and I. Oshri, "Social ties, knowledge sharing and successful collaboration in globally distributed system development projects," *European Journal of Information Systems,* vol. 14, no. 1, 2005.

[8] P. J. Hinds, and S. P. Weisband, "Knowledge sharing and shared understanding in virtual teams," *Virtual teams that work: Creating conditions for virtual team effectiveness*, pp. 21-36, 2003.

[9] I. Rus, and M. Lindvall, "Knowledge management in software engineering," *Software, IEEE,* vol. 19, no. 3, pp. 26-38, 2002.

[10] A. Aurum, F. Daneshgar, and J. Ward, "Investigating Knowledge Management practices in software development organisations - An Australian experience," *Information and Software Technology,* vol. 50, no. 6, pp. 511-533, 2008.

[11] I. Nonaka, "A Dynamic Theory of Organizational Knowledge Creation," *Organization Science,* vol. 5, no. 1, pp. 14-37, 1994.

[12] M. Hansen, N. Nohria, and T. Tierney, "What is your strategy for managing knowledge," *The knowledge management yearbook,* vol. 2001, pp. 55-69, 2000.

[13] C. Manteli, B. van den Hooff, A. Tang, and H. van Vliet, "The Impact of Multi-site Software Governance on Knowledge Management." pp. 40-49.

[14] A. Boden, G. Avram, L. Bannon, and V. Wulf, "Knowledge Management in Distributed Software Development Teams - Does Culture Matter?." pp. 18-27.

[15] A. Davis, and D. Khazanchi, "Does mutual knowledge affect virtual team performance? Theoretical analysis and anecdotal evidence," *American Journal of Business,* vol. 22, no. 2, pp. 57-66, 2007.

[16] P. J. Hinds, and M. Mortensen, "Understanding conflict in geographically distributed teams: The moderating effects of shared identity, shared context, and spontaneous communication," *Organization science,* vol. 16, no. 3, pp. 290-307, 2005.

[17] I. Oshri, J. Kotlarsky, and L. Willcocks, "Missing links: building critical social ties for global collaborative teamwork," *Commun. ACM,* vol. 51, no. 4, pp. 76-81, 2008.

[18] M. Hertzum, "Six roles of documents in professional's work." pp. 41-60.

[19] J. L. Krein, P. Wagstrom, S. M. S. Jr, C. Williams, and C. D. Knutson, "The Problem of Private Information in Large Software Organizations," in ICSSP, 2011, pp. 218-222.

[20] H. Tellioglu, and I. Wagner, "Software cultures," *Communications of the ACM,* vol. 42, no. 12, pp. 71-77, 1999.

[21] M. Cataldo, M. Bass, J. D. Herbsleb, and L. Bass, "On Coordination Mechanisms in Global Software Development." pp. 71-80.

[22] R. Yin, *Case study research: Design and methods*: Sage Publications, Inc, 2003.

[23] V. Braun, and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology,* vol. 3, no. 2, 2006.

[24] J. M. Corbin, and A. Strauss., "Grounded theory research: Procedures, canons, and evaluative criteria," *Qualitative Sociology*, no. 13, 1990.

[25] B. Curtis, H. Krasner, and N. Iscoe, "A field study of the software design process for large systems," *Commun. ACM,* vol. 31, no. 11, pp. 1268-1287, 1988.

[26] C. D. Cramton, and K. L. Orvis, "Overcoming barriers to information sharing in virtual teams," *Virtual teams that work: Creating conditions for virtual team effectiveness*, pp. 214-230, 2003.