Technical Report 12-08:
Knowledge Management in Distributed Agile Software Development

Siva Dorairaj, James Noble and Petra Malik

February 21, 2012

Abstract

Software development teams need highly valuable knowledge to carry out knowledge-intensive development activities. Agile teams are cross-functional teams that promote sharing of project-specific knowledge through frequent face-to-face interaction, effective communication and customer collaboration. Knowledge sharing is difficult for distributed Agile teams due to spatial, temporal, and cultural barriers, which negatively affect face-to-face interaction, communication and collaboration. There seems to be very few studies that focus on knowledge management in distributed Agile teams. Through a Grounded Theory study that involved 45 participants from 28 different software companies in the USA, India and Australia, we investigate distributed software development from the specific perspective of Agile teams. In this paper, we describe how Agile teams gather, store, share and use knowledge in distributed software development.

1 Introduction

Software development is a series of knowledge-intensive activities that encompasses gathering requirements, analyzing problems, designing, coding, testing, and ensuring the software remain up-to-date and bug-free [12, 45]. Software development teams need highly valuable knowledge relating to project management, software development and its processes, and technologies to be successful in their projects. A traditional software development team carries out software development activities through specific knowledge-rich roles such as Project Managers, Software Analysts, Designers, Developers, or Testers, that are associated with specific stages in the development process [14]. An Agile software development team is a cross-functional team without specific roles given to its members [8]. The concept of 'cross-functional team' does not mean 'collection of cross-functional individuals' even though members of an Agile team often strive to be 'omni-knowledgeable' [51, 75]. Cross-functional team just means that the team as a whole has all skills needed to build the software, and that every team member is willing to coordinate, communicate and collaborate to get the job done [41, 53]. Therefore, Agile teams carry out the software development activities through effective communication and customer collaboration [8]. Team members frequently share complex and context-specific knowledge that are essential to deliver business value to the customers [48]. From this perspective, effective knowledge sharing in the team is imperative for the success of Agile projects [50]. Several studies, however, point out that knowledge sharing is difficult for distributed Agile teams due the challenges in communication, particularly face-to-face interaction between team members in different locations [10, 42].

This raises a critical question: *How do Agile teams gather, store, share, and use knowledge in distributed software development?* We found the answer to this question through a Grounded Theory study that involved 45 participants from 28 different software companies in the USA, Australia and India. As we investigate distributed software development from the perspective of Agile practitioners, several practices that promote effective knowledge management in the distributed teams emerged from the analysis. We describe these practices through the knowledge management processes: *knowledge generation*, *knowledge codification*, *knowledge transfer*, and *knowledge application.*

2 Background

Software companies are increasingly venturing into distributed software development. Besides lowering development costs, one of the key reasons for organizing a distributed team is to benefit from the superior knowledge that resides in remote locations [37, 46]. Agile teams strive to develop software with less cost and with a higher quality while embracing change throughout the software development process [1, 66]. Self-managing Agile teams need to leverage knowledge, techniques, and tools to manage the development of software products. Learning and acquiring knowledge is critical in Agile software development that focuses on customer satisfaction through continuous delivery of valuable software [38]. Value of the knowledge increases within the team or organization when knowledge has a key purpose, and focuses on mission and core values [4, 67]. Software development teams, however, face difficulties to gain clear understanding with respect to knowledge and managing knowledge [59]. Though several studies [21, 43, 77] have defined knowledge, the following is one of the most common definition:

> "Knowledge is a fluid mix of framed experience, values, contextual information, and expert insight that provides a framework for evaluating and incorporating new experiences and information. It originates and is applied in the minds of the knowers. In organizations, it often becomes imbedded not only in documents or repositories but also in organizational routines, processes, practices, and norms." [21, p.5]

Knowledge is divided into two types: tacit and explicit [56]. *Tacit knowledge* is the action-oriented or "know how" knowledge that guides human behaviour but it cannot be openly expressed, whereas *explicit knowledge* is the academic knowledge or "know what" knowledge that can represented in written or verbal forms [67]. A wealth of research literature points out that the most valuable knowledge within an organization is essentially tacit. This tacit knowledge is embodied in people in the form of experience and skills, and in the processes and products that people create. Nonaka and Takeuchi [54], writing about knowledge creating companies, describe that "although we use the term 'organizational' knowledge creation, the organization cannot create knowledge on its own without the initiative of the individual and the interaction that takes place within the group." This means that organization create knowledge from the tacit knowledge of individuals in the organization. Knowledge Management (KM) describes the processes through which organizations manage tacit and explicit knowledge held within the individuals of the organizations to achieve competitive advantage [4, 62].

KM is "a method that simplifies the process of sharing, distributing, creating, capturing and understanding of a company's knowledge" [21]. The approach to knowledge management often assumes that the knowledge within an organization will largely consist of tacit knowledge that remains in the heads of individuals in the organization. Wilson [77] suggests that "whenever we wish to express what we know, we can only do so by uttering messages of one kind or another – oral, written, graphic, gestural or even through body language". Davenport and Prusak [21] explain that 'knowledge exists within people, part and parcel of human complexity and unpredictability', and therefore knowledge transmitted outside of the mind becomes mere information for the receiver. From this perspective we understand organizations can offer tools, techniques, processes and procedures, but managing knowledge still will remain the responsibility of the individual.

Though early KM models focused on explicit and quantifiable knowledge, recent studies on KM focus on the collective capabilities of the individuals in an organization such as accumulation of experience, learning of new behaviours and understandings, and solving of work-context problems [20]. There are several different models for KM, "allowing researchers and practitioners to generalise, communicate and apply the findings" [60]. Within a KM model, there are four knowledge processes: *knowledge generation* (creation and acquisition), *knowledge codification* (storing), *knowledge transfer* (sharing), and *knowledge application* (use) [4, 21, 69].

Knowledge generation is creating innovation and opportunities for resolution of problems, and acquiring knowledge from external sources [21]. Knowledge codification is translation of tacit knowledge into explicit knowledge in written or verbal forms for storage in repositories [21]. Knowledge transfer is sharing of knowledge between individuals within an organization [21]. Finally, knowledge application is using knowledge to gain the competitive advantage [21, 4]. Software companies often exhort managers to manage knowledge in the routine and practices that the software development teams transforms into valuable product and services. There is, however, a paucity of studies into knowledge management in distributed Agile teams [11, 42]. To address this gap, we investigate how distributed teams create and acquire, store, share and use knowledge from the specific perspective of Agile teams.

3 Research Method

Grounded Theory (GT) is a systematic research method that emphasises the generation of theory derived from systematic and rigorous analysis of data. We chose GT as our research method for two main reasons. Firstly, GT is suitable to be used in areas that are under-explored or where a new perspective might be beneficial [63], and the literature on distributed Agile software development, particularly on knowledge management, is still scarce [42]. Secondly, GT allows researchers to study social interactions and the behaviour of people [35], and Agile methods focus on people and their interactions in software development teams [64]. GT is being used successfully to study the social nature of Agile teams [30, 40, 48, 57, 75]. Using Glaser's guidelines, our study started with a general area of interest – distributed Agile software development – rather than beginning with a specific research question because formulating a research question before commencing the study can lead to preconceived ideas of the research phenomenon [17, 33, 36, 39, 52]. This does not mean that there is no specific problem for the research, but rather the problems and its key concerns will emerge in the initial stages of data analysis [32, 33].

Table 1: Summary of Participant and Distributed Agile Project. (Agile Position: Scrum Master (SM), Agile Coach (AC), Developer (DEV), Business Analyst (BA), Quality Analyst (QA), Product Owner (PO), Senior Management (MGT))

Participant (code)	**Agile Position**	**Project Distribution**	**Project Domain**	**Team Size**	**Duration (months)**	**Iteration (weeks)**
P1	DEV	USA-India	Financial Services	8 to 10	10	2
P2	AC	USA-India	E-Commerce	12 to 14	12	2
P3	SM	USA-Western Europe-India	Mobile Application	10	8	3
P4	AC	USA-China	Online Trading	10	8	2
P5	AC	USA-India	Internet Media Services	8	12	2 to 3
P6	DEV	USA-UK	Internet Hosting Services	20 to 22	8	2
P7	AC	USA-Argentina-India	Internet Domain Services	18	6	2
P8	DEV	USA-Australia-India	Publishing	9 to 10	8	2
P9	DEV	Western Europe-Brazil	Web Search Engine	14	24	2 to 3
P10	SM	USA-Argentina-India	Software Platform	10 to 12	8	3
P11	SM	USA-Middle East-India	Web Services	13	10	2
P12	DEV	USA-India	Internet Hosting Services	12	18	2
P13	SM	USA-India	Web Portal	17 to 20	5	2
P14	DEV	USA-India	E-Commerce	16 to 17	36	2
P15	QA	USA-India	E-Commerce	16	18	2
P16	SM	USA-India	E-Commerce	16	18	2
P17	DEV	USA-India	E-Commerce	16	18	2
P18	BA	UK-India	Financial Services	8	12	2
P19	DEV	USA-India	Insurance	8 to 10	10	3
P20	MGT	Australia-India	E-Commerce	9 to 12	12	2 to 3
P21	SM	USA-Australia	Financial Services	15	9	2
P22	SM	Australia-India	E Commerce	9 to 12	12	2 to 3
P23	QA	Japan-India-China	Power Distribution	7 to 8	4	2
P24	AC	Western Europe-India	Automobile	9	5	2
P25	SM	USA-India	Information Security	24	6	3
P26	AC	USA-India	Healthcare	16	ongoing	3
P27	SM	USA-Brazil	Finacial Services	30	6	2
P28	MGT	USA-India	E-Commerce	20	18	3
P29	SM	USA-India	Social Networking	14	10	2
P30	AC	Western Europe-India	Retail	8 to 10	ongoing	2 to 3
P31	AC	UK-India	Retail	15 to 20	ongoing	3
P32	MGT	UK-South Africa	Retail	12	18	2
P33	AC	Australia-Ukraine-India	Recruitment	50	24	3
P34	AC	USA-India	Real Estate	6 to 8	10	2
P35	AC	USA-India	Online Payment	8	18	3
P36	QA	Canada-India	Web Services	10 to 15	18	2
P37	DEV	Western Europe-India	E-Commerce	16	4	2
P38	BA	USA-India	E-Commerce	28	ongoing	2
P39	AC	USA-India	Telecommunication	22 to 25	6 to 7	2
P40	DEV	Australia-India	Online Trading	7	6	1
P41	AC	Northern Europe-India	Retail	10 to 12	ongoing	2
P42	MGT	USA-India	Healthcare	7	ongoing	4
P43	SM	USA-India	E-Commerce	7	ongoing	2 to 3
P44	PO	Canada-India	Cloud Computing	10 to 12	ongoing	2 to 3
P45	MGT	USA-India	Financial Services	10	ongoing	3

3.1 Context

This study involved 45 Agile practitioners from 28 different software companies in the USA, India and Australia. Participants fulfilled the specified criteria of (1) at least four years of Agile software development experience, and (2) direct involvement in a current or recent distributed Agile project either in a technical, business or management role. Table 1 shows the participant and project details. Our participants adopted Agile methods, primarily Scrum and XP, in distributed software development. The project distribution varied from 2 to 4 countries, often across several time zones and different cultures. The project durations varied from 6 to 24 months, and some projects were still ongoing during data collection. Though the projects started with a small team size, some teams had scaled through having team of teams of up to 50 members to accommodate the increasing complexity of the projects.

We realize that Agile development teams are cross-functional teams with one common 'Team' role and formed of generalizing specialists that have different functional skills within the team itself. Yet, the development team members specified a particular position in the projects (e.g. Developer, Quality Analyst, or Business Analyst) based on the position held in the organization, or the key responsibilities that were carried out in the team. We also included Scrum Master, Agile Coach, Product Owner, and Senior Management (e.g. Development Manager, Recruitment Director, and Director of Technology) in order to get a rounded perspective of the research phenomenon. Due to privacy and ethical consideration, we will only identify our participants using the codes P1 to P45. All data were personally collected and analysed by the primary researcher in order to maintain consistency in the application of GT.

3.2 Data Collection and Analysis

We conducted face-to-face, one-on-one interviews with our participants using open-ended questions. The interviews were voice recorded with the consent from the participants so that we could concentrate on the conversation. Initially a set of interview questions was prepared to develop a smooth discussion with the participants. The interview questions mainly focused on the challenges faced by distributed Agile teams and the strategies adopted by teams to overcome these challenges.

Data collection and analysis occurs simultaneously in a GT study. We analysed interviews during and after each interview. We particularly avoided collecting all the data during a specific data collection phase, and then analysing them in a subsequent data analysis phase. Several key concerns emerged from the initial analysis: *knowledge sharing*, *trust*, *communication*, *culture*, and *team interaction*. We formulated research questions based on the emergent key concerns and investigated every one of them. The ongoing analysis guided the future questions, and also the choice of future participants.

When an interview has been transcribed, the transcript was analyzed line-by-line using open coding to explore the meaning in the data [5, 31]. Open coding breaks down, examines, compares, conceptualizes and categorizes the data [71]. We collated key points from the data and assigned a *code*, or a summary phrase, to each key point. Using GT's *constant comparison method* [34], we constantly compared each code with the codes from the same interview, and those from other interviews. The codes that were related to a common theme were grouped together to produce a second level of abstraction called a *concept*.

As we continuously compared the codes, many fresh concepts emerged. These concepts were analysed using constant comparison method to produce a third level of abstraction called a *category.* We wrote-up memos on the ideas about the codes, concepts and categories, and their inter-relationships with one another.

We presented several other findings from this study in different papers [22, 23, 24, 25, 26, 27]. In this paper, we aimed to investigate the question described in section 1, which is *"How do Agile teams gather, store, share, and use knowledge in distributed software development?"* As we analyzed the data, theoretical sensitivity on the area of knowledge management from different disciplines guided us to consolidate the emergent concepts based on the four knowledge management processes: *knowledge generation*, *knowledge codification*, *knowledge transfer*, and *knowledge application*, and the category as *knowledge management in distributed Agile teams.* In the next section, we describe the findings using these concepts and category that emerged, and explicate the research phenomenon using the memos that were written during analysis.

4 Results

In this section, we describe the practices that promote effective management of knowledge in the distributed Agile teams through the four processes of knowledge management: *Knowledge Generation*, *Knowledge Codification*, *Knowledge Transfer*, and *Knowledge Application.* We present selected quotations drawn from our interviews that had lead us to the emergence of these concepts and category.

4.1 Knowledge Generation

Knowledge generation is divided into two main subprocesses: *knowledge creation* and *knowledge acquisition.* Knowledge can either be generated through original knowledge that exists within the individuals in the organization, or knowledge can be acquired from external sources and integrated into the organization.

Inception: Project inception is an iterative series of structured workshops that provides opportunities for stakeholders to crystallize their ideas in collaboration with development teams [68]. An inception concludes with a shared vision for the project and prepares the development team for Agile development iterations [18].

The stakeholders engage in discussions that includes project goals and vision, business benefits, costs, schedule, success measures, technology assumptions, and rules of engagement. The knowledge gained from the inception is crucial to get a project started the right path with a shared understanding:

> "*Whenever we start a project, we start it with an inception workshop [where] we met up with the clients, talked about the project goals, their vision of the project, and what is the scope of the project. We decided on the communication plan, the overlap hours between the different locations, how the requirements are going to be passed to the project team, who are the Product Owners, who are the stakeholders, and who is going to do the sign-offs.*" – P11, Scrum Master.

Customer collaboration: Responsibilities of a customer includes driving the software development project, providing project requirements and performing acceptance testing [47]. Customer collaborates with project teams to enhance business-technical collaboration on a project and sets the direction of the project to determine what and when to build. [48].

Customer collaboration improves communication with development teams and enhance knowledge creation through active discussion on project requirements and feedback on the features:

> "*We have discussion with the customers on a daily basis, and involved the customers throughout the development phases. So the [project] teams get the requirements and enough explanation from them on what we need to develop.*" – P24, Agile Coach.

Formal training: Through formal training sessions managers are able to standardize training content and practices across multiple teams in an organization [15]. The training sessions are largely beneficial to disseminate process and technical knowledge to novices, particularly on the domain and system requirements, and the technologies that are used in the projects [15].

Formal training programmes that include soft-skills, technical skills and Agile practices have been found useful to upgrade the knowledge and skills of team members. Hiring professional trainers and experienced consultants, and sending teams for training sessions and certifications can be expensive and time consuming, and therefore management support is required to design the training programmes and implement them across the organization:

> "*We have a schedule of training programmes that everybody in the organization needs to attend, particularly a set of 'soft-skills' courses such as personality development, communication, management training and leadership. And from the technical side, we train them on the process, the Agile practices and what we do in the project team. We put in the formal training programme to make sure that everybody upgrades their knowledge and skills, and knows what the practice is.*" – P45, Senior Management.

Communities of practice: Communities of practice are self-organizing groups that consist of individuals who share information, insight, experience, and technical skills on a specialised discipline, and collaborate on common challenges or stimulating new ideas [49]. Communities of practice focus on learning through sharing knowledge and developing a set of good practices, and provide opportunities to leverage the tacit knowledge within the community members [42, 44].

Team members who are open minded to accept suggestion and recommendations from outside the project team participate in activities within communities of practice to gain a useful perspective on knowing and learning, and to enhance collaboration between team members and other members of the communities:

> "*We have a lot of community activities going on within [our organization] and across India. We frequently meet up to discuss the issues faced in different projects, and receive suggestions or possible solutions. So a broad level of knowledge sharing within the communities of practice is brought across multiple projects.*" – P36, Quality Analyst.

Self-learning: Learning is an important aspect of KM that encompasses what should be learned, when it should be learned, and who should be learning it [28]. Individuals in a team should realize that knowledge value degrades with time, and that evaluation of what knowledge is needed when to meet the needs of the project activities is essential for the success of the project [28]. Managers understand that individuals seek different knowledge at different time, and therefore encouraging the individuals for self-learning and organizing learning opportunities for them are sensible management responsibilities [28].

Self-managing Agile teams engage in self-learning when a specific knowledge seems to be important and required for project activities. Learning can happen through interacting and collaborating with team members, accessing and understanding information available in knowledge repositories, or participating in community activities:

> "*We encourage self-learning in our team. We practice an 'open culture' [where] we practice a lot of 'peer learning' and 'community learning'. Basically people are smart, knowledgeable and have access to information, and people learn from the wealth of interaction in the team.*" – P44, Product Owner.

4.2 Knowledge Codification

Knowledge codification is the translation of tacit knowledge into explicit knowledge in written, visual or verbal format for storage within knowledge management systems that provide efficient mechanisms to access the codified knowledge [21].

Wiki: A Wiki is a collection of web pages with several special publishing and collaboration features that significantly improve conversational knowledge creation and sharing [73]. Earlier knowledge management conversational technologies such as e-mails, audio and video conferencing, and Instant Messaging allow individuals to create and share knowledge through web based discussion forums and online communities. A Wiki offers a model for high impact knowledge management because it provide opportunities for collaborative knowledge creation in the team and facilitate individual learning [19].

An individual initially provides information and reflects his own knowledge into a Wiki article. Other members of the team can extend the individual's knowledge through detailed description, intellectual discussions, or reflection of experiences into the Wiki article in a form that maps the initial knowledge:

> "*We enter technical details of every feature, even a bug, into the Wiki. So when a new member joins the team, we can provide all the details that are available, and the [team member] can search for the features, review the documents, add comments, and discuss with other members. Wiki is one tool that helped with knowledge sharing and collaboration.*" – P33, Agile Coach.

Documents: The purpose of documentation is to describe the requirements, design, implementation and limitation of the system to those who are responsible for using, maintaining and evolving the system [65]. Design documents that describe the architectural specifications of the higher-level structure and behavior of system are crucial for modern software that are complex and frequently revisited for upgrades [65, 61].

Agile methods prefer “working software over comprehensive documentation” [29]. The effort of creating documents describing planned implementation and keeping them up-to-date in Agile projects that “welcome changing requirements, even late in development” may not be worth of time and cost [42]. Though Agile methods reduce the emphasis on documentation, team members write the design documents or test cases especially upon requests from customers. Agile teams do not suggest that documents are obsolete but change the focus when to create and maintain them:

> “*To some extend we explain on the design choices and test cases in the code itself, [but] we write separate documentation when the client manager or client architect request for them.*” – P19, Developer.

Technical presentation: Sharing of valuable ideas and concepts, and technical expertise through short presentations can be challenging for the presenter. The information shared needs to be interesting and memorable to the audience for effective sharing. The content of the presentation can be captured and stored into a knowledge management system to allow team members to revisit and understand further the presentation at a later time:

> “*We have a 15 minutes presentation once in a week so that team members are able to share their own technical expertise with the team. So what happens is that all these presentations were collected and put in a shared location for the team members to view and understand. This collection [of presentations] really helped them in the development of technical features.*” – P34, Agile Coach.

4.3 Knowledge Transfer

Knowledge transfer is movement of knowledge from sender to receiver, understanding the knowledge transmitted and integrating it with existing knowledge within the receiver’s mind [4]. The transfer of knowledge does not guarantee a full replication of the knowledge for the receiver. In fact, knowledge is often modified in the mind of the receiver.

Daily Scrum: Daily Scrum is the daily meetings that brings team members together to synchronize the works of all team members, identify possible impediments to progress in projects, and schedule other meetings [64]. When the meeting involves all members, including remote members in a distributed team, it serves as a team-building activity that makes the members feel part of the team [58].

During the daily meetings that often lasts just for fifteen minutes, team members need to focus on the information shared to the team. A daily meeting can fulfill its purpose only when meaningful information that promotes understanding and learning is shared to the team:

> “*I have seen so many dysfunctional stand-up meetings, and the reason is always that there are people in the stand-up meetings that whatever information shared is not meaningful for the teams. So the important thing [that] is the information is valuable for everyone on that daily meeting.*” – P4, Agile Coach.

Inception: At initial stage of a project, a small team is formed to assess the viability of engaging in the project. The team that consists of Project Managers, Technical Leads, and Business Analysts, participates in the project inception and documents the outcome of discussions with stakeholders. If the project kicks-off, the team members who attended the inception shares complete knowledge about the project to the rest of the team members:

> "*When I came back from the inception, I helped to wrap up to the entire team, and give complete knowledge about the analysis of the inception.*" – P11, Scrum Master.

Pair programming: Pair programming promotes integration of collaborative knowledge within the pair programmers, and provides an effective means of knowledge dissemination and retention in an organization [55]. Benefits of pair programming includes increased knowledge transfer, enhanced learning, effective knowledge creation [76]. Hence, pair programmers produce better program than solo programmers [70].

Pair programmers mutually transfer knowledge through effective collaboration and frequent communication. Teams ideally pair a junior member with a senior member to encourage the junior member to learn particularly performance impacts in programming:

> "*We normally get the junior developers to pair with the seniors. Though the junior developers are really brilliant, they just don't have the experience and the exposure to understand performance impacts.*" – P12, Developer .

Tools: Considerable amount of explicit knowledge is readily available on the Internet, and therefore additional support in the form of knowledge management tools that promote the use of the information sources and integrating knowledge management into the development process can be beneficial for the team members [13, 42]. Customer representatives often provide information on requirements through verbal communication to individuals who requested for the additional information from time to time [74]. Therefore, tools are particularly important for distributed teams to conserve and distribute knowledge gathered by team members in different locations, and to share project status and velocity:

> "*It certainly helps to have good tools that reflect team's progress, the current [project] status and priorities, and use the knowledge on the web. Tools can make it easier and effective to share that information with people in remote locations.*" – P20, Senior Management.

Visits: Though knowledge transfer can happen during daily meetings through technology-mediated communication such audio or video conference, face-to-face interaction is prefered, especially when there are high levels of complexity and ambiguity within the projects [50]. Lack of non-verbal cues such as facial expression or hand gestures in technology-mediated communication reduces the richness of the information exchanged between the distributed team members [26].

Distributed team members visit other members in different locations when and as needed to gain better understanding of critical situations through face-to-face interactions that offer rich communication and effective knowledge sharing:

> "*Right now something critical is going on, and we need some sort of knowledge from the [customer]. Two people from India went to the [customer's] site to work through with all those concerns. We learn a lot of things from their site and we benefit from this knowledge transfer.* " – P26, Agile Coach.

Rotation: The on-site team members often gets to learn the business and domain knowledge much faster than offshore team members due to frequent face-to-face interaction with Product Owner or customer representatives [72]. Rotation of team members between the onsite and offshore locations, often between 3-6 months, promotes the distribution of the business and domain knowledge across the teams:

> "*The primary reason for rotation would be knowledge sharing. Often somebody has got some specific knowledge that need to be shared, or somebody wants to learn a specific knowledge available in one location. So we have some people coming over [to India], or we have some people going and working with the team in that location for some time and then bring back that knowledge.*" – P27, Scrum Master.

On-site customer: On-site customers drive the software development by continually providing correct and complete understanding of their needs, and effectively contributing to the growth and utilization of domain concerned knowledge [9].

When customers are working on-site with the team, collaboration can be enhanced through effective participation in release planning, daily meetings, review meetings and retropectives. These on-site customers provide timely feedback throughout the iterative development process:

> "*There was one on-site customer sitting together with the team. The customer was working very collaboratively along with us throughout the iterations and review meetings, and giving their feedback along the way.* " – P27, Scrum Master.

Cross-functional teams: A cross-functional team as a whole has skilled individuals performing all required roles to build a software through effective communication and collaboration [15]. Cross-functional teams focus on the whole solution rather, and not just the software development process.

A cross-functional team is ideally staffed with developers, analysts, testers, and individuals from other disciplines that can contribute to the success of a project through active knowledge sharing within the teams:

> "*Within the team we have cross-functional members as one team. There were three Developers, two Business Analysts, one Project Manager, one Accounts Manager, and two customer representatives in our team. We don't have role-specific separations. So this was a reason for our successful Agile implementation.* " – P24, Agile Coach.

Discussions: Discussions facilitate openness and communication between team members in different locations. Discussions with subject matter experts on specific issues faced in the teams provide opportunities to refine, reprioritize, and generate requirements and solution [16].

When distributed teams engaged in frequent discussions, particularly with subject matter experts, team members were able to gain knowledge on the Agile practices, technologies and other processes within the projects:

"*We introduced 'bi-weekly technical huddle' sessions to share technical knowledge on what we are doing and what are the things blocking our work. We have Skype conversations with the onsite team and client, and discuss things with subject matter experts.*" – P38, Business Analyst.

4.4 Knowledge Application

Knowledge application is using knowledge to create and sustain competitive advantage in organizations [4]. Knowledge that is available within a specific team or general organization ideally should be accessible in a way that promotes effective use of the knowledge.

Similar Context: When a team member interacts with the Wiki, knowledge and information are interchanged between the member and the Wiki. The new knowledge that the team member can infer out of the information in the collaboratively and iteratively improved Wiki articles represents collaborative knowledge [19]. This new knowledge can be integrated into the individual's existing knowledge and used effectively in similar project-context activities:

"*When you go through that Wiki forum, you get to see the notes [that] people have posted. So, in many other projects we use these notes that comes with some detailed explanations associated to the issues. People are able to use the knowledge from that [Wiki] forum in similar context.*" – P17, Developer.

Problem solving: Although technology can assist people with storage and transfer of knowledge artifacts, the knowledge itself can be created and exist only in human's mind [21, 77]. Therefore, team members need to consolidate and understand related information contained in knowledge documents such as Wiki, and create a new knowledge that can help them to realize a possible solution for a problem that arise in their projects:

"*The knowledge sharing help them to gain more knowledge, and when some problems arise, an idea strikes and they are able to deliver a solution.*" – P29, Scrum Master.

Future sprints/projects: Knowledge from multiple technologies and functional domains is a neccesity for complex projects. Project teams ideally should have access to information in knowledge repositories within and outside the organizations for successful completion of projects [74].

One of the 12 principles behind the Agile Manifesto describes the nature of Agile teams that 'the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly' [29]. In this context, Agile teams conduct sprint reviews and sprint retrospectives after each sprint to assess the success of the sprint against the sprint goals, and seek for improvement opportunities and new ideas. Knowledge acquired during the course of a project, particularly during the sprint reviews and retropectives, is very useful to promote continuous improvement to the process and practice that is in place within the team:

"*After each iteration, we are doing the review meetings and retrospectives. Based on that feedback, we'll be taking corrective actions and analyzing whether the project is going in the right way and whether it can be completed in time. So, we plan [future] iterations and projects accordingly.*" – P23, Quality Analyst.

5 Related Work

There is a wealth of literature on knowledge management from different fields such as Organizational Theory, Psychology, Sociology, Information System, and Computer Science. In this section we discuss several studies that investigate the knowledge management in software development, particularly in Agile software development.

The concept of *Experience Factory* in Software Engineering describes that a software development project can improve its performance in terms of cost, schedule and quality by leveraging experience from previous projects [7]. The *Experience Factory* collects experience from software development projects, analyzes and synthesizes all kinds of experience, including lessons learned, project data, and technical reports, stores the integrated experience, and supplies the 'emergent' knowledge to the current projects on demand [7, 59].

Through an empirical study that investigated the effects of knowledge delivery factors on software development efficiency, Ajile and Sun [3] report that software development projects using the 'pull' approach for knowledge delivery will be more effective than projects using 'push' approach. The 'pull' approach expects users to search for knowledge by themselves based on what kind knowledge are required to get the job done, whereas the 'push' approach aims at proactively providing knowledge to users. From our study, we found that knowledge-oriented team members are encouraged to search for knowledge from different sources such as Wiki, community events or technical blogs, and gather knowledge through effective communication and frequent interaction among team members. As face-to-face interaction can be difficult for distributed teams, efforts are made using technology-mediated communication for self-learning and interactive peer-learning between team members from different locations.

Melnik and Maurer [50] discuss the role of social conversation and frequent interaction as the key element of effective knowledge sharing in Agile process. They suggest that verbal face-to-face interaction facilitates effective knowledge sharing and achieving higher velocity accomplishments, and results in the success of the software development projects. Though face-to-face interaction is affected in distributed software development, our participants recognize the importance of face-to-face interaction for effective knowledge sharing. Our findings indicate that team members travel to different locations on a regular basis, and team members in certain roles such as Developers or Quality Analysts rotate between different locations and work with other members of the team for a specific period of time. Travelling and rotation provide opportunities for team members to meet face-to-face, and the team bonding that was built while working together continue to facilitate knowledge sharing even when team members are no longer working in the same location.

Through an empirical study that compared the knowledge creation process in an XP project and a traditional project, Bahli and Zeid [6] found that software development using extreme programming (XP) enhances the creation of tacit knowledge among team members. The adoption of XP was facilitated by a high degree of knowledge creation through frequent interaction among team member, and effective collaboration and communication between the development team and the users. From our study, we found that active collaboration among the development teams members and effective discussions with the customer representatives on a daily basis throughout the development phases enhance the knowledge creation in distributed Agile teams.

Chau and Maurer[14] describe that Agile teams share knowledge through several practices: release and sprint planning, pair-programming, on-site customers, cross-functional teams, daily Scrum meetings, and project retrospectives. They argue that these practices are team-oriented and rely on face-to-face interaction between team members. Hence, these practices are effective for collocated and small teams, and do not facilitate inter-team learning or knowledge sharing in distributed Agile teams. We found that sprint planning, daily Scrum, sprint review and project retrospective meetings were in practice. The distributed Agile teams were cross-functional, and team members at different locations share knowledge through effective use of technology-mediated communication such as video-conferencing and knowledge management tools such as Wiki. Collaboration through distributed pair-programming and periodic interactive technical discourses were useful for knowledge sharing in distrbuted teams.

6 Limitation

The inherent limitation of a Grounded Theory study is that the findings are grounded in the specific contexts explored in the research [2, 39]. These contexts were dictated by the availability of the Agile practitioners to participate in this study, and by our choice of research destinations. We do not claim that our findings are universally generalisable to all distributed Agile projects, but rather our findings accurately characterise the contexts studied.

7 Conclusions

Through a Grounded Theory study, we investigate distributed software development from the specific perspective of Agile practitioners. Several key practices of knowledge management in distributed software development emerged from our analysis. We describe these practices through the four processes of knowledge management: *knowledge generation*, *knowledge codification*, *knowledge transfer*, and *knowledge application*. Agile methods promotes knowledge sharing through coordination, communication and collaboration between team members. Scrum [64] advocates knowledge sharing through its four process activities: sprint planning, daily Scrum, sprint reviews and sprint retrospectives. XP [8] encourages knowledge sharing particularly through release and iteration planning, pair programming, and on-site customers. Agile teams are cross-functional teams that the team as a whole has the collection of skills required to perform software development activities and deliver business values to customers.

Knowledge required for Agile software development is very context-dependent, and often it is difficult to transfer the context to different projects even within the same organization. Therefore, critically analyzing the knowledge before reuse is crucial for succesful implementation of knowledge management systems. Knowledge generation involves creation and acquisition of knowledge through several practices: project inception, customer collaboration, formal training sessions, communities of practice and self-learning. Knowledge codification translates tacit knowledge to explicit knowledge that are stored as Wiki articles, design documents, and presentation materials in knowledge management systems. We found that transfer of knowledge between distributed team members occurs, using suitable tools, during daily Scrum, inception workshops, visits to different sites, pair-programming, rotation and discussion sessions. The shared and stored knowledge is integrated into an individual's existing knowledge to create new knowledge that is used in similar project-context software development and problem solving activities, and to gain competitive advantage in future sprints or projects.

We hope that realization of these practices can contribute to successful implementation of distributed Agile projects. We acknowledge that there can be other knowledge management practices that can be useful and effective in their own contexts, but did not emerge from our analysis. Future studies can assess the effectiveness of each of these practices in different project contexts.

8 ACKNOWLEDGMENTS

Thanks to the Agile practitioners who participated in this study. This study is supported by Universiti Tenaga Nasional (Malaysia) Ph.D. scholarship.

References

[1] N. Abbas, A. M. Gravell, and G. B. Wills. Historical roots of Agile methods: Where did Agile thinking come from? In *Agile Processes in Software Engineering and Extreme Programming*, volume 9 of *Lecture Notes in Business Information Processing*, pages 94–103. Springer Berlin Heidelberg, 2008.

[2] S. Adolph, W. Hall, and P. Kruchten. A methodological leg to stand on: Lessons learned using grounded theory to study software development. In *Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research*, pages 166–178, New York, NY, USA, 2008. ACM.

[3] S. A. Ajila and Z. Sun. Knowledge management: Impact of knowledge delivery factors on software product development efficiency. In *Proceedings of the IEEE International Conference on Information Reuse and Integration*, pages 320–325, Nov. 2004.

[4] M. Alavi and D. E. Leidner. Knowledge management systems: Issues, challenges and benefits. *Communications of the Association for Information Systems*, 1, February 1999.

[5] G. Allan. The use of grounded theory as a research method: warts all. In *European Conference on Research Methodology for Business and Management Studies*, pages 9–19. MCIL, 2005.

[6] B. Bahli and E. S. A. Zeid. The role of knowledge creation in adopting extreme programming model: An empirical study. In *Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference on Information and Communications Technology.*, pages 75–87, Dec. 2005.

[7] V. R. Basili, G. Caldiera, and H. D. Rombach. *Experience Factory.* John Wiley & Sons, Inc., 1994.

[8] K. Beck. *Extreme Programming Explained: Embrace Change.* Addison-Wesley, Upper Saddle River, 2000.

[9] H. Beyer, K. Holtzblatt, and L. Baker. An Agile customer-centered method: Rapid contextual design. In C. Zannier, H. Erdogmus, and L. Lindstrom, editors, *Extreme Programming and Agile Methods - XP/Agile Universe 2004*, volume 3134 of *Lecture Notes in Computer Science*, pages 527–554. Springer Berlin / Heidelberg, 2004.

[10] A. Boden and G. Avram. Bridging knowledge distribution - the role of knowledge brokers in distributed software development teams. In *ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, pages 8–11, May 2009.

[11] A. Boden, G. Avram, L. Bannon, and V. Wulf. Knowledge management in distributed software development teams - does culture matter? In *Fourth IEEE International Conference on Global Software Engineering, ICGSE 2009.*, pages 18 –27, July 2009.

[12] B. W. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61–72, May 1988.

[13] S. Bowen and F. Maurer. Process support and knowledge management for virtual teams doing Agile software development. In *Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International*, pages 1118–1120, 2002.

[14] T. Chau and F. Maurer. Knowledge sharing in Agile software teams. In W. Lenski, editor, *Logic versus Approximation*, volume 3075 of *Lecture Notes in Computer Science*, pages 173–183. Springer Berlin / Heidelberg, 2004.

[15] T. Chau, F. Maurer, and G. Melnik. Knowledge sharing: Agile methods vs. Tayloristic methods. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*, pages 302–307, June 2003.

[16] S. Cohan. Successful customer collaboration resulting in the right product for the end user. In *AGILE Conference*, pages 284–288, 2008.

[17] G. Coleman and R. O'Connor. Using grounded theory to understand software process improvement: A study of Irish software product companies. *Information and Software Technology*, 49(6):654–667, 2007.

[18] M. Coram and S. Bohner. The impact of Agile methods on software project management. In *The 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pages 363–370, April 2005.

[19] U. Cress and J. Kimmerle. A systemic and cognitive view on collaborative knowledge building with wikis. *The International Journal of Computer-Supported Collaborative Learning*, 3:105–122, 2008.

[20] M. Curado and I. Ramos. Knowledge management in organizations: A new proposal. In *Proceedings of 11th European Conference on Knowledge Management*, pages 323–333, Universidade Lusada de Vila Nova de Famalico, Famalico, Portugal, 2010.

[21] T. H. Davenport and L. Prusak. *Working Knowledge: How Organizations Manage What They Know.* Harvard Business School Press, Boston, MA, USA, 1998.

[22] S. Dorairaj, J. Noble, and P. Malik. Understanding the importance of trust in distributed Agile projects: A practical perspective. In A. Sillitti, A. Martin, X. Wang, and E. Whitworth, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 48 of *Lecture Notes in Business Information Processing*, pages 172–177. Springer Berlin Heidelberg, Trondheim, Norway, 2010.

[23] S. Dorairaj, J. Noble, and P. Malik. Bridging cultural differences: A grounded theory perspective. In *Proceedings of the 4th India Software Engineering Conference*, ISEC '11, pages 3–10, New York, NY, USA, 2011. ACM.

[24] S. Dorairaj, J. Noble, and P. Malik. Effective communication in distributed Agile software development teams. In A. Sillitti, O. Hazzan, E. Bache, X. Albaladejo, W. Aalst, J. Mylopoulos, M. Rosemann, M. J. Shaw, and C. Szyperski, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 77 of *Lecture Notes in Business Information Processing*, pages 102–116. Springer Berlin Heidelberg, 2011.

[25] S. Dorairaj, J. Noble, and P. Malik. Distribution and Agility: It's all about trust. Technical Report ECSTR-12-01, Victoria University of Wellington, New Zealand, January 2012.

[26] S. Dorairaj, J. Noble, and P. Malik. Understanding lack of trust in distributed agile teams: A grounded theory study. Technical Report ECSTR-12-07, Victoria University of Wellington, New Zealand, February 2012.

[27] S. Dorairaj, J. Noble, and P. Malik. Understanding team dynamics in distributed Agile software development. Technical Report ECSTR-12-02, Victoria University of Wellington, New Zealand, January 2012.

[28] R. Dove. Knowledge mangement, response ability, and the agile enterprise. *Journal of Knowledge Management*, 3(1):18–35, 1999.

[29] M. Fowler and J. Highsmith. The agile manifesto. *Software Development*, 9(8):28–35, 2001.

[30] D. Fox, J. Sillito, and F. Maurer. Agile methods and user-centered design: How these two methodologies are being successfully integrated in industry. In *Agile 2008 Conference*, pages 63–72, 2008.

[31] S. Georgieva and G. Allan. Best practices in project management through a grounded theory lens. *Electronic Journal of Business Research Methods*, 6(1):43–52, 2008.

[32] B. Glaser. *Basics of Grounded Theory Analysis: Emergence vs Forcing.* Sociology Press, Mill Valley, CA, 1992.

[33] B. Glaser. *Doing Grounded Theory: Issues and Discussions.* Sociology Press, Mill Valley, CA, 1998.

[34] B. G. Glaser. The constant comparative method of qualitative analysis. *Social Problems*, 12(4):436–445, 1965.

[35] B. G. Glaser and A. L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research.* Sociology Press, Aldine, Chicago, 1967.

[36] C. Goulding. Grounded theory: some reflections on paradigm, procedures and misconceptions. Working Paper WP006/99, University of Wolverhampton, UK, June 1999.

[37] J. D. Herbsleb and D. Moitra. Global software development. *IEEE Software*, 18(2):16–20, 2001.

[38] J. Highsmith. Messy, exciting, and anxiety-ridden: Adaptive software development. 10(1), 1997.

[39] R. Hoda, J. Noble, and S. Marshall. Developing a grounded theory to explain the practices of self-organizing Agile teams. *Empirical Software Engineering*, pages 1–31.

[40] R. Hoda, J. Noble, and S. Marshall. Organizing self-organizing teams. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, pages 285–294, New York, USA, 2010.

[41] S. Hole and N. B. Moe. A case study of coordination in distributed Agile software development. In R. V. O'Connor, N. Baddoo, K. Smolander, and R. Messnarz, editors, *Software Process Improvement*, volume 16 of *Communications in Computer and Information Science*, pages 189–200. Springer Berlin Heidelberg, 2008.

[42] H. Holz and F. Maurer. Knowledge management support for distributed Agile software processes. In S. Henninger and F. Maurer, editors, *Advances in Learning Software Organizations*, volume 2640 of *Lecture Notes in Computer Science*, pages 60–80. Springer Berlin / Heidelberg, 2003.

[43] P. Iske and T. Boekhoff. The value of knowledge doesnt́ exist: A framework for valuing the potential of knowledge. In D. Karagiannis and U. Reimer, editors, *Practical Aspects of Knowledge Management*, volume 2569 of *Lecture Notes in Computer Science*, pages 632–638. Springer Berlin / Heidelberg, 2002.

[44] T. Kahkonen. Agile methods for large organizations - building communities of practice. In *Agile Development Conference, 2004*, pages 2–10, June 2004.

[45] C. Larman and V. R. Basili. Iterative and incremental developments: A brief history. *Computer*, 36(6):47–56, June 2003.

[46] L. Layman, L. Williams, D. Damian, and H. Bures. Essential communication practices for Extreme Programming in a global software development team. *Information and Software Technology*, 48(9):781–794, 2006. Special Issue Section: Distributed Software Development.

[47] A. Martin, R. Biddle, and J. Noble. The XP customer role in practice: Three studies. In *Proceedings of the Agile Development Conference*, pages 42–54, Washington DC, USA, 2004. IEEE Computer Society.

[48] A. Martin, R. Biddle, and J. Noble. The XP customer team: A grounded theory. In *Proceedings of the AGILE*, pages 57–64, 2009.

[49] R. McDermott. Learning across teams: The role of communities of practice in team organization. *Knowledge Management Review*, 2, 1999.

[50] G. Melnik and F. Maurer. Direct verbal communication as a catalyst of Agile knowledge sharing. In *Proceedings of the Agile Development Conference*, pages 21–31, June 2004.

[51] N. B. Moe, T. Dingsyr, and T. Dyb. A teamwork model for understanding an Agile team: A case study of a Scrum project. *Information and Software Technology*, 52(5):480–491, 2010.

[52] A. Moghaddam. Coding issues in grounded theory. *Issues In Educational Research*, 16:52–66, 2006.

[53] S. Nerur, R. Mahapatra, and G. Mangalaraj. Challenges of migrating to Agile methodologies. *Communications of ACM*, 48:72–78, May 2005.

[54] I. Nonaka and H. Takeuchi. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation.* Oxford University Press, Inc., New York, USA, 1995.

[55] D. W. Palmieri. Knowledge management through pair programming. Master's thesis, North Carolina State University, 2002.

[56] M. Polanyi. *The Tacit Dimension.* Doubleday & Company, Inc., Garden City, New York, USA, 1966.

[57] B. Ramesh, L. Cao, K. Mohan, and P. Xu. Can distributed software development be Agile? *Communication of the ACM*, 49(10):41–46, 2006.

[58] L. Rising and N. S. Janoff. The scrum software development process for small teams. *Software, IEEE*, 17(4):26–32, July/Aug 2000.

[59] I. Rus and M. Lindvall. Knowledge management in software engineering. *IEEE Software*, 19(3):26–38, May/June 2002.

[60] A. Sarmento, I. Ramos, J. A. Carvalho, F. Lopes, and P. Morais. A research approach classification for knowledge management. San Diego, California, USA, 2005.

[61] T. Sauer. Using design rationales for Agile documentation. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*, pages 326–331, June 2003.

[62] J. R. Schermerhorn. *Management.* John Wiley & Sons, 10 edition, 2009.

[63] R. S. Schreiber and P. N. Stern. *Using Grounded Theory in Nursing.* Springer Publishing, Broadway, New York, 2001.

[64] K. Schwaber and M. Beedle. *Agile Software Development with Scrum.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[65] B. Selic. Agile documentation, anyone? *IEEE Software*, 26(6):11–12, Nov-Dec 2009.

[66] A. Sidky, J. Arthur, and S. Bohner. A disciplined approach to adopting Agile practices: The Agile adoption framework. *Innovations in Systems and Software Engineering*, 3:203–216, 2007.

[67] E. A. Smith. The role of tacit and explicit knowledge in the workplace. *Journal of Knowledge Management*, 5:311–321, 2001.

[68] J. Smith. An approach to developing a performance brief at the project inception stage. *Architectural Engineering and Design Management*, 1(1):3–20, 2005.

[69] S. Staab, R. Studer, H.-P. Schnurr, and Y. Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1):26–34, Jan/Feb 2001.

[70] D. Stotts, L. Williams, N. Nagappan, P. Baheti, D. Jen, and A. Jackson. Virtual teaming: Experiments and experiences with distributed pair programming. In F. Maurer and D. Wells, editors, *Extreme Programming and Agile Methods - XP/Agile Universe 2003*, volume 2753 of *Lecture Notes in Computer Science*, pages 129–141. Springer Berlin / Heidelberg, 2003.

[71] A. Strauss and J. Corbin. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory.* Sage Publications, 1998.

[72] K. Sureshchandra and J. Shrinivasavadhani. Adopting Agile in distributed development. In *Proceedings of the 2008 IEEE International Conference on Global Software Engineering*, pages 217–221, Washington, DC, USA, 2008. IEEE Computer Society.

[73] C. Wagner. Wiki: A technology for conversational knowledge management and group collaboration. *Communications of the Association for Information Systems*, 13:265–289, 2004.

[74] D. B. Walz, J. J. Elam, and B. Curtis. Inside a software design team: Knowledge acquisition, sharing, and integration. *Communications of ACM*, 36:63–77, October 1993.

[75] E. Whitworth and R. Biddle. The social nature of Agile teams. In *Proceedings of the AGILE*, pages 26–36, Washington, DC, USA, 2007. IEEE Computer Society.

[76] L. Williams, A. Shukla, and A. I. Anton. An initial exploration of the relationship between pair programming and Brooks' law. In *Agile Development Conference, 2004*, pages 11–20, June 2004.

[77] T. Wilson. The nonsense of knowledge management. *Information Research*, 8(1), October 2002.