Effective Communication in Distributed Agile Software Development Teams

Siva Dorairaj, James Noble, and Petra Malik

School of Engineering and Computer Science,
Victoria University of Wellington,
Wellington, New Zealand
{siva.dorairaj,james.noble,petra.malik}@ecs.vuw.ac.nz

Abstract. Agile methods prefer team members to be collocated to promote effective communication between team members. Effective communication is crucial for distributed Agile software development where team members are scattered across different geographic locations, and often across several time zones. We are conducting in a Grounded Theory study that explores distributed Agile software development from the perspective of Agile practitioners. We present the causes of communication challenges, and the strategies adopted by our participants to overcome communication challenges in distributed Agile teams.

Keywords: Agile Methods, Grounded Theory, Distributed Software Development, Communication.

1 Introduction

Effective communication is important for software development teams to facilitate knowledge transfer rapidly between team members, to allow team members to understand the requirements from clients, and to help team members perform development activities efficiently [1, 2]. Realising the benefits of effective communication, Agile methods prefer team members to be collocated to promote effective communication through frequent interactions between team members. [3]. Moreover, one of the 12 principles behind the Agile Manifesto asserts that face-to-face conversation is the most efficient and effective method of conveying information to and within a development team [4]. Several studies, however, indicate that communication is the main challenge for distributed software development where team members are scattered across different geographic locations, and often across several time zones [1, 5, 6, 7, 8].

In this paper we present the findings of an ongoing Grounded Theory study that explores distributed Agile software development from the perspective of Agile practitioners. This study involved 18 Agile practitioners from 10 different software companies in the USA and India. We analyse the causes of communication challenges, and present the practical strategies adopted by

A. Sillitti et al. (Eds.): XP 2011, LNBIP 77, pp. 102–116, 2011.

Agile practitioners to overcome the communication challenges in distributed Agile software development.

The rest of the paper is structured as follows: section 2 describes the research method, data collection and data analysis; sections 3 describes the context of the study; sections 4 and 5 present the results of the study; section 6 describes several related work; section 7 describes limitations of our study, followed by conclusion in section 8.

2 Research Method

Grounded Theory (GT) is an inductive research method originally developed by Barney G. Glaser and Anslem L. Strauss [9]. The Grounded Theory Institute [10] defines GT as "*the systematic generation of theory from systematic research*". GT emphasises the generation of a *grounded theory* regarding an area of interest. The term "*grounded theory*" refers to a theory that is developed inductively from a corpus of data. GT researchers gather data, particularly qualitative data from interviews and observations, and systematically discover a theory derived directly from the data.

We chose GT as our research method for several reasons. Firstly, GT is suitable to be used in areas that are under-explored or where a new perspective might be beneficial [11], and the literature on distributed Agile software development is scarce [5, 12]. Secondly, GT allows researchers to study social interactions and behaviour of people in the context of solving problems [9], and Agile methods focus on people and their interactions in the teams. Thirdly, GT is increasingly being used successfully to study the social nature of Agile teams [13, 14, 15]. GT allows a theory to emerge from systematic analysis of qualitative data from practitioners of Agile software development.

A GT research project begins with a general area of interest, and not with specific research questions, because defining research questions leads to a preconceived conceptual description [16]. This does not mean that there is no specific problem for the research, but rather the problem and its concerns will emerge in the initial stages of data analysis. [16, 17]. Using Glaser's approach, we started out our research with a general area of interest — distributed Agile software development. As we progressed through data collection and data analysis, several concerns have emerged. We investigated further on several concerns, and reported our findings [18, 19]. Further progress in data analysis gave rise to the research questions: "What are the causes of communication challenges in distributed Agile teams?", and "What are the strategies adopted by Agile practitioners to overcome communication challenges faced by the distributed teams?". We continued data collection by focusing on our research questions. Grounded Theory gives us the capability to answer these questions. In order to maintain consistency in the application of Grounded Theory, all data was collected and analysed personally by the primary researcher.

2.1 Data Collection

Our data collection technique was interviewing the Agile practitioners. We wanted to acquire our participant's experience in regard to distributed Agile software development. Initially, we emailed the Agile practitioners informing them about our research and gain their consent for an interview. We then scheduled the interviews for an hour at a mutually agreed location. We conducted face-to-face, one-on-one interviews with our participants using open-ended questions. Face-to-face interviews provided us with the opportunity not only to gather verbal information but also to observe the body language of the participants during the conversations. The interviews were voice-recorded with the consent from the participants. Although Glaser [16] advises against recording interviews, we find it convenient to maintain a record of interviews, and to analyse the data at ease.

We had prepared an interview guide with a set of guiding questions. We commenced the interview by asking the participants about their experience, and their roles and responsibilities in the distributed Agile projects. We asked the participants to highlight the challenges they faced in distributed Agile projects, and the strategies adopted by their team to manage their projects effectively. We phrased our questions cautiously so that the issues in distributed Agile software development would emerge from the participants rather than from our own agenda. The ongoing analysis during the interview reveals participant's concerns. We gradually focused our attention on these concerns rather than our prepared questions.

2.2 Data Analysis

Data collection and data analysis occurs simultaneously in a GT study. We avoided collecting all the data during a specific data collection phase, and then analysing them in a subsequent data analysis phase. Rather, we analysed interviews during and after each interview. The key concerns that emerged from the ongoing data analysis enabled adjustments to the inquiry by including these concerns in subsequent interviews. The developing theory guided the future interviews, and the choice of future participants. Voice recording the interviews helped us to concentrate on the conversation and conduct continuous analysis during the interview.

We transcribed each interview and reviewed the transcripts line-by-line to explore the meaning in the data by searching for similarities and differences. We collated key points from the data and assigned a *code*, a summary phrase, to each key point. This is the *coding* [9] process that involves categorisation, interpretation and analysis of the data. As we identified codes, we constantly compared each code with the codes from the same interview, and those from other interviews. This is called the *constant comparison method* [20]. The codes that related to a common theme were grouped together to produce a second level of abstractions called a *concept*. As we continuously compare the codes, many fresh concepts emerge. These concepts were analysed using constant comparison to produce a third level of abstractions called a *category*.

As a result of the analysis, the concepts *Time Zone Factor*, *Lack of Communication Tools*, *Language Barriers*, and *Lack of Teamwork* gave rise to the category *Lack of Effective Communication*.

Another set of concepts uncovered from the analysis were *Reducing Time Zone Factor*, *Leveraging Communication Tools and Techniques*, *Addressing Language Barriers*, *Developing Trusted Relationships*, *Increasing Effective Formal Communication*, and *Increasing Effective Informal Communication*. From these concepts, the category *Increasing Effective Communication* had emerged.

2.3 The Emergence of Theory

The ongoing study requires the data collection and data analysis to be continued until *theoretical saturation* [9] is attained – that is when no more new concepts or categories emerge from the data. We will continuously write-up memos on the ideas about the codes, concepts and categories, and their inter-relationships with one another. Glaser [16] describes that this *theoretical memoing* is the "core stage" of a Grounded Theory study. The collection of the theoretical memos developed during the data analysis helps to generate the theory. *Sorting* the theoretical memos is an "essential step" carried out to help formulate the theory [21]. These sorted memos are then weaved together to explicate the research phenomenon. This explication is called a *grounded theory* — a theory that is truly grounded in the data. The grounded theory generated from the sorted memos richly explicates the research phenomenon, and exhibits a strong connections between the concepts and categories [21, 17].

3 Context

We interviewed 18 Agile practitioners from 10 different software organisations in the USA and India to which we had access. All the participants we interviewed have adopted Agile methods, primarily Scrum and XP, in their distributed software development projects. All the participants have at least 4 years of experience in distributed Agile software development projects. Moreover, all the participants have experience in collocated Agile projects prior to their involvement in the distributed Agile projects. In order to acquire a comprehensive understanding of the concerns in distributed Agile software development, we had interviewed participants from a range of different roles within the distributed Agile projects. In particular, we interviewed Scrum Masters, Agile Coaches, Developers, Application Testers, and Business Analysts.

Table 1 shows the summary of the distributed Agile projects investigated in our study. The project distribution varied from 2 to 3 countries, the project durations varied from 6 to 24 months, and the team size varied from 8 to 22 people on different projects. Due to privacy and ethical consideration, we will only identify the distributed Agile software development projects using the codes C1 to C14, and our participants using the codes P1 to P18.

Table 1. Summary of Distributed Agile Projects. (Agile Role: Developer(Dev), Agile Coach(AC), Scrum Master (SM), Application Tester (AT), Business Analyst(BA)).

Project (code)	Participant (code)	Agile Role	Project Distribution	Agile Method	Team Size	Project Duration (months)
C1	P1	Dev	USA-India	Scrum	8 to 10	10
C2	P2	AC	USA-India	Scrum & XP	12 to 14	12
C3	P3	SM	USA-Western Europe-India	Scrum	10	8
C4	P4	AC	USA-China	Scrum & XP	10	8
C5	P5	AC	USA-India	Scrum & XP	8	12
C6	P6	Dev	USA-UK	Scrum & XP	20 to 22	8
C7	P7	AC	USA-Argentina-India	Scrum	18	6
C8	P8	Dev	USA-Australia-India	Scrum & XP	9 to 10	8
C9	P9	Dev	Brazil-Western Europe	Scrum & Lean	14	24
C10	P10	SM	USA-Argentina-India	Scrum	10 to 12	8
C11	P11	SM	USA-Middle East-India	Scrum & XP	13	10
C12	P12	Dev	USA-India	Scrum & XP	12	18
C12	P13	Dev	USA-India	Scrum & XP	12	18
C12	P14	Dev	USA-India	Scrum & XP	12	18
C13	P15	AT	USA-India	Scrum & XP	16	18
C13	P16	SM	USA-India	Scrum & XP	16	18
C13	P17	Dev	USA-India	Scrum & XP	16	18
C14	P18	BA	UK-India	Scrum & XP	8	12

4 Causes of Lack of Effective Communication

In this section, we describe the causes of the communication challenges in distributed Agile software development derived from the concepts *Time Zone*, *Lack of Communication Tools*, *Language Barriers*, and *Lack of Teamwork* that gave rise to the category *Lack of Effective Communication*. We present selected quotations drawn from our interviews that lead us to the emergence of the category *Lack of Effective Communication*.

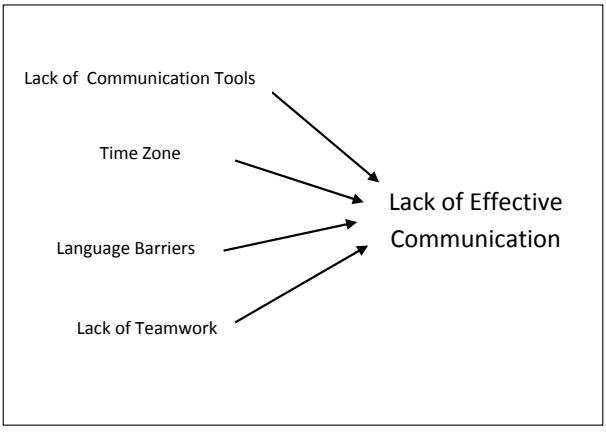

Fig. 1. Emergence of category *Lack of Effective Communication*

Figure 1 shows the concepts that gave rise to the category *Lack of Effective Communication*.

4.1 Time Zone

Due to difference in time zones, communication between the globally distributed development team members is difficult:

> "*We are in central time zone; we were off by 12 hours. We were holding up our daily calls.*" —P1, Developer.

> "*The time difference between west coast [of the USA and India] is 12.5 hours. Communication among [distributed] team members can be so difficult.*" —P16, Scrum Master.

Time zone differences provides minimum communication opportunity to distributed team members:

> "*They are doing the calls in the morning when the India folks are at home. Those India QC folks are going to have only 45 minutes communication with the USA folks onshore.*" —P1, Developer.

Scheduling group meetings outside normal working hours can be difficult during certain time of the year:

> "*In the winter it was very hard to have meetings outside office hours because of shorter time [overlaps].*" —P9, Developer.

Time zone difficulties are common is many distributed Agile projects because the client companies are frequently located in developed countries (e.g the USA, or the UK), and the offshore teams are located in developing countries (e.g. India, or Latin America).

4.2 Lack of Communication Tools

Distributed Agile projects may suffer if the distributed teams do not have suitable communication tools:

> "*And this [distributed] project failed because we did not have the right tools in place. We did not have any video conference. We should have used more video or voice instead of written communication.*" —P9, Developer.

Appropriate communication tools are important to help the team communicate effectively:

> "*It is much harder [to communicate] in a group meeting using the phone. So [the team] should use Web based conferencing.*" —P6, Developer.

Ideally, teams should have access to different communication tools and use them as necessary.

4.3 Language Barriers

Agile practitioners whose native language is not English often face language barriers when communicating in English with team members:

> "*Meetings were [conducted] in English, [but] English was not the main language for everyone.*" —P7, Agile Coach.

> "[*There] were four developers, [and] only one had a good [command of] English.*" —P9, Developer.

Language barriers can exist even among the Agile practitioners who have good command of English:

> "*When [the Indian team members] speak English, they would try to speak faster than what is comfortable for them because their understanding is that Americans talk fast and they too talk fast.*"—P1, Developer.

Language barriers can limit communication in the team:

> "*The meeting was longer [than usual], and because of that [language barrier], they did not talk much. Some people took a long time to think the idea and express [it] properly in English.*" —P7, Agile Coach.

Language barriers are common in many distributed Agile projects because a particular language (often English) is used to communicate between distributed Agile teams although part of the team are not native speakers of that language.

4.4 Lack of Teamwork

All team members should ideally communicate with all other team members, and not just with several selected people in a team:

> "*Our only communication that time was with the software architect, and that was not enough. We did not have direct communication with the rest of the team.*" —P9, Developer.

Overall communication is affected when some team members do not wish to contribute to the team's effort to communicate to each other:

> "*Most people would be like "Leave me alone! I do not want to be known." That was a big challenge for us because our [distributed] team needed visibility, [and] we needed everyone to communicate in the team.*" —P8, Developer.

> "*In the current project, the Indian folks haven't even been engaged on the [phone] calls.*" —P1, Developer.

All the team members should understand the importance of teamwork in a distributed software development, and contribute to the entire team's effort to promote effective communication.

5 Strategies for Increasing Effective Communication

In this section, we describe the strategies adopted by our participants to overcome the challenges faced by distributed Agile teams. These strategies are derived from the concepts *Reducing Time Zone*, *Leveraging Communication Tools and Techniques*, *Addressing Language Barriers*, *Developing Trusted Relationships*, *Increasing Effective Formal Communication*, and *Increasing Effective Informal Communication that gave rise to the category Increasing Effective Communication. We present selected quotations drawn from our interviews that had lead us to the emergence of the category Increasing Effective Communication.*

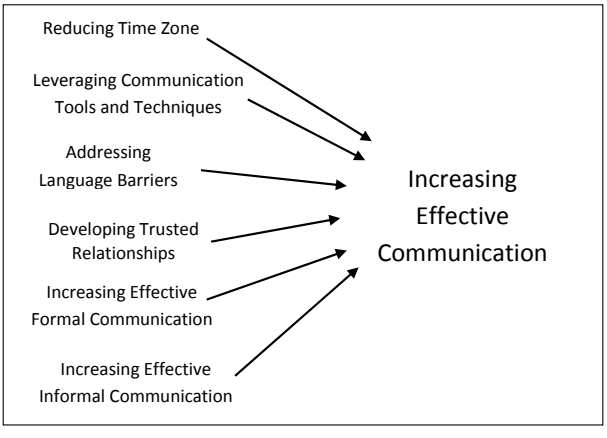

Fig. 2. Emergence of category *Increasing Effective Communication*

Figure 2 shows the concepts that gave rise to the category *Increasing Effective Communication.*

5.1 Reducing Time Zone

Our participants realise that team members should be ideally distributed between countries that have smaller time difference to minimise communication difficulty:

> "*The best way is to have the whole team in the same time zone instead of splitting these teams across different time zones. The time zone is painful!* " —P5, Agile Coach.

> "*It is always better that [the teams] are closer together, in terms of physical distance and also in time difference. It would be ideal to work with people from the same time zone.*" —P4, Agile Coach.

Agile methods prefer teams to be collocated. If the Agile teams are distributed, the time zone should be ideally reduced to minimise communication difficulties.

5.2 Leveraging Communication Tools and Techniques

Our practitioners realise that face-to-face communication is the best way to communicate effectively. Distributed teams should be ideally provided with video, voice or text communication options to address the communication challenges faced in distributed software development:

> "*Face-to-face communication is still the best! Team members [need to] leverage tools to the best extend to increase the communication. Teams that are across geographical boundaries use tools like instant messaging, email, video conferencing, web-based backlog management tools.*" —P5, Agile Coach.

In a distributed team, face-to-face meetings are expensive and difficult to schedule. Video-conferencing is often used as an alternative technique to capture the visual aspect of communication, such as body language, allowing the team to communicate effectively:

> "*When we started using video conferencing, we found the visual aspect in communication is so important that we encourage teams to, at least once or twice a month, have video conferencing.*" —P1, Developer.

The distributed team should be ideally ready to use different tools and techniques at the spur of the moment. Writing or drawing on a board and combining video communication can increase communication for distributed teams:

> "*When we start to get into intense discussion, we often go into Skype. As I am talking, I will be writing and drawing [on the board]. They can hear it and read at the same time. You get the web-cam to point to the whiteboard and say, "Look, this is what I mean!*" " —P3, Scrum Master.

Our participants (P1, P2, P3, P4, P5, P6, P9) explain that leveraging communication tools and techniques promoted effective communication in their distributed teams.

5.3 Addressing Language Barriers

Agile practitioners can address the language barriers by consciously speaking in a manner that promotes clear communication:

> "*We tell [the Indian team] to speak slowly and deliberately. Their English is much better and communication got much better.*" —P1, Developer.

> "*When we are having an audio conference, and if someone's accent is different, we may not be able to understand everything. So, we try to talk slower than usual.*" —P14, Developer.

Prior to a scheduled meeting, the onsite and offshore team members should casually discuss about the matters that will be brought up in the meeting. This will give them the opportunity to express themselves clearly, and team members can understand each other well during the meeting:

> "*Before we have the daily Scrum meetings, the offshore team sends an email answering basic Scrum questions - what did I do yesterday, what am I going to do today, what are my problems and impediments - so that they can have it in writing [before the meeting].*" —P3, Scrum Master.

Our participants (P1, P3, P7, P9, P14) realise that the concerns arising from language barriers must be addresed as early as possible to improve effective communication and to build good relationships in the distributed teams.

5.4 Developing Trusted Relationships

Our participants describe that developing team relationships is important to communicate effectively with the team members in distributed software development:

> "*The key to that [effective communication] was good [team] relationships. It is [about] building [team] relationships and rapport with the people before you get distributed to different countries. You have to learn to make friends with people that you are working with.*" —P11, Scrum Master.

When team members have developed trusted relationships, it is much easier for the team to communicate with each other:

> "*By having enough trust, you know [that] you are not going to offend somebody in some manner while having discussions.*" —P1, Developer.

Trusted relationship helps the distributed team members to interact with people in the teams:

> "*If you do not know someone, you blame him, but if you know him, you try to understand what he does. Knowing the people [in the team] helps me to talk to people, [and] interact with people.*" —P9, Developer.

Trust provides a significant bond in team relationships. Our participants find that the distributed team members interact and communicate frequently once the trusted relationships have been developed.

5.5 Increasing Effective Formal Communication

Formal communication includes official meetings during inceptions, formal weekly meetings and daily standups, and specification documents from clients. The customer and development team should ideally meet face-to-face during an inception phase to discuss the project goals, and gain commitment from the team to produce value to the project:

> "*One really ideal time [to communicate] is during inception. It is very critical to have the whole team there [at inception] because that is where the shared understanding of the entire project is established. That is so critical to the success of the project. An inception is useful for the product owner to crystallise their idea in front of the team, in collaboration with the team.*" —P2, Agile Coach.

Due to budget constraints, sometimes only the key stakeholders attend the inception meeting to discuss the significant business and requirements risks which must be addressed before the project kickoff. The outcome of the meeting should be clearly explained to the rest of the team.

> "*When we had the inception, we met up with the client and talked about the goal of the project, and their vision, [and] what is the scope. When I came back, I helped to rap up the team, and give knowledge about the analysis of the inception.*" —P11, Scrum Master.

Daily stand-up meetings between distributed team members are important because these meetings help the team to respond quickly to the changes in the project:

> "*I have seen so many dysfunctional stand-up meetings and the reason is always that there are people in the stand up meeting sharing information that is not meaningful for others. So the important thing is that the information is valuable for everyone on that daily meetings. They have to be aware [that] information is important.*" —P4, Agile Coach.

Team members should realise that it is possible to get quick responses from the stakeholders during stand-up meetings. Team members should prepare to communicate effectively during the stand-up meetings:

> "*During the stand-up, all the client party will be there, and that will be the best time for me [to ask questions]. They may discuss among themselves and respond. I'll prepare the most important question to ask in stand-ups. I feel this is more effective than emails.*" —P15, Application Tester.

> "*People [in the team] can communicate with each other at any point of time either through phone or video. We would have two daily stand-ups because it helps people to talk.*" —P8, Developer.

Our participants (P1, P2, P4, P8, P11, P15) feel that team members should communicate effectively during all scheduled meetings.

5.6 Increasing Effective Informal Communication

Informal communication includes face-to-face conversation, casual communication through video-conferencing or telephone, written communication through email, online chat and short message service (SMS) which are not formally scheduled by the teams. Informal communication can increase knowledge transfer between people in the distributed teams and build trust.

Our participant realise that effective informal communication helps to develop a strong team:

> "*. . . the first 15 minutes [of video conferencing] was open time and you could talk about anything you want. And, that is when we started seeing a very strong team building and that became probably the strongest thing we did as far as building team.*" —P1, Developer.

Casual conversations amongst team members helps to increase understanding of their software development project:

> "*Teams that are motivated to make it [distributed software development] work, can use all the means of communication. They often casually talk on phone to know things better.*" —P6, Developer.

Informal communication does not occur frequently when team members are distributed mainly due to time zone difference and language barriers. Team that deliberately create the opportunity to communicate casually are able to develop strong teams.

6 Related Work

Paasivaara and Lassenius [12] discuss potential challenges in adopting Agile methods in globally distributed software development. The researchers recognise that the application of Agile methods to globally distributed software development poses severe communication challenges. In our study, we found that distributed Agile teams were facing communication challenges due to lack of communication tools, time zone, language barriers, and lack of team work.

Vax and Michaud [22] describe the potential challenges in distributed software development such as the inability to meet face-to-face on daily basis, language and cultural barriers, and lack of trust. The researchers explain that adopting Agile methods to distributed projects can be challenging but by constructing the right team with the right skills and expertise, and effectively leveraging tools and techniques, the teams can acquire tremendous benefits in terms of scalability, productivity, cost management, risk reduction and improved software quality. The researchers address the challenges derived from distributed teams by building trust, developing effective communication plan, sharing electronic workspace, reducing time zones and conducting frequent retrospective.

Young and Terashima [23] recognise that having distributed Agile teams present many challenges that slowed down software development processes. The researchers describe the strategies adopted by distributed Agile teams to overcome time zones problems and cultures differences. In order to deliver successful software releases, the distributed teams deliberately made conscious efforts to improve communication, built strong working relationships amongst the distributed team members, and ensured that the software architecture was suitable for all teams. In our study, we found that our participants overcome communication challenges by reducing time zones, leveraging communication tools and techniques, addressing language barriers, developing trusted relationships, and increasing formal and informal communication in distributed Agile teams.

Korkala and Abrahamsson [5] recognise that distributed software development is increasingly becoming important for software companies. The researchers explain that distributed software development is already burdened with several challenges, and Agile methods bring further challenges in the form of their reliance on informal communication and volatile requirements. The researchers

describe that the high volatile requirements in Agile software development are managed through effective communication. In our study, we found that increasing effective communication helps to resolve communication challenges in distributed Agile teams.

7 Limitation

An inherent limitation of doing Grounded Theory study is that the findings are grounded in the specific contexts explored in the research. These contexts were dictated by our choice of research destination which was limited to 10 software development projects the USA and India. We do not claim that our findings are generally applicable to all distributed Agile software development projects, but rather our findings accurately characterize the contexts studied [24].

8 Conclusion

Our research explored distributed Agile software development from the perspective of the Agile practitioners. We interviewed 18 Agile practitioners from 10 different software organisations in the USA and India. The results of our Grounded Theory study reveal that distributed Agile teams face communication challenges caused by the time zone, lack of communication tools, language barriers, and lack of teamwork. The participants adopted several practical strategies to overcome communication challenges in distributed Agile software development by reducing time zone, leveraging communication tools and techniques, addressing language barriers, developing trusted relationships, increasing formal communication, and increasing informal communication. Participants found the strategies to be largely useful and effective in their own contexts. Future studies could explore the viability of these strategies in different contexts such as with distributed teams from other countries and cultures.

Acknowledgments. Thanks are due to all Agile practitioners who participated in this research. This research is supported by Universiti Tenaga Nasional (Malaysia) PhD scholarship.

References

1. Herbsleb, J.D., Mockus, A.: An empirical study of speed and communication in globally distributed software development. IEEE Transactions on Software Engineering 29(6), 481–494 (2003)
2. Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., Still, J.: The impact of Agile practices on communication in software development. Empirical Software Engineering 13, 303–337 (2008)
3. Cockburn, A.: Agile Software Development. Addison-Wesley, Indianapolis (2002)

4. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: Manifesto for Agile Software Development, http://www.agilemanifesto.org/principles.html (last accessed on February 16, 2011)
5. Korkala, M., Abrahamsson, P.: Communication in distributed Agile development: A case study. In: 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 203–210 (2007)
6. Korkala, M., Pikkarainen, M., Conboy, K.: Distributed agile development: A case study of customer communication challenges. In: Abrahamsson, P., Marchesi, M., Maurer, F. (eds.) XP 2009. LNBIP, vol. 31, pp. 161–167. Springer, Heidelberg (2009)
7. Mockus, A., Herbsleb, J.D.: Challenges of global software development. In: Proceedings of the Seventh International Software Metrics Symposium, pp. 182–184 (2001)
8. Prikladnicki, R., Audy, J.L.N., Damian, D., de Oliveira, T.C.: Distributed software development: Practices and challenges in different business strategies of offshoring and onshoring. In: International Conference on Global Software Engineering, pp. 262–274 (2007)
9. Glaser, B.G., Strauss, A.L.: The Discovery of Grounded Theory: Strategies for Qualitative Research. Sociology Press, Aldine (1967)
10. Phine, J.: Grounded Theory Institute. The Grounded Theory methodology of Barney G. Glaser, http://www.groundedtheory.com (last accessed on February 16, 2011)
11. Schreiber, R.S., Stern, P.N.: Using Grounded Theory in Nursing. Springer Publishing, Broadway (2001)
12. Paasivaara, M., Lassenius, C.: Could global software development benefit from Agile methods? In: Proceedings of the IEEE International Conference on Global Software Engineering, pp. 109–113. IEEE Computer Society, Washington, DC (2006)
13. Hoda, R., Noble, J., Marshall, S.: Balancing acts: Walking the Agile tightrope. In: Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, pp. 5–12. ACM, New York (2010)
14. Martin, A., Biddle, R., Noble, J.: The XP customer team: A grounded theory. In: Proceedings of the AGILE, pp. 57–64 (2009)
15. Whitworth, E., Biddle, R.: The social nature of Agile teams. In: Proceedings of the AGILE, pp. 26–36. IEEE Computer Society, Washington, DC (2007)
16. Glaser, B.: Doing Grounded Theory: Issues and Discussions. Sociology Press, Mill Valley (1998)
17. Glaser, B.: Basics of Grounded Theory Analysis: Emergence vs Forcing. Sociology Press, Mill Valley (1992)
18. Dorairaj, S., Noble, J., Malik, P.: Understanding the importance of trust in distributed agile projects: A practical perspective. In: Sillitti, A., Martin, A., Wang, X., Whitworth, E. (eds.) XP 2010. LNBIP, vol. 48, pp. 172–177. Springer, Heidelberg (2010)
19. Dorairaj, S., Noble, J., Malik, P.: Bridging cultural differences: A grounded theory perspective. In: Proceedings of the 4th India Software Engineering Conference, Thiruvananthapuram, Kerala, India (2011) (to be published)
20. Glaser, B.G.: The constant comparative method of qualitative analysis. Social Problems 12(4), 436–445 (1965)

21. Glaser, B.: Theoritical Sensitivity: Advances in Methodology of Grounded Theory. Sociology Press, Mill Valley (1978)
22. Vax, M., Michaud, S.: Distributed Agile: Growing a practice together. In: Proceedings of the AGILE, pp. 310–314. IEEE Computer Society, Los Alamitos (2008)
23. Young, C., Terashima, H.: How did we adapt Agile processes to our distributed development? In: Proceedings of the AGILE, pp. 304–309. IEEE Computer Society, Los Alamitos (2008)
24. Adolph, S., Hall, W., Kruchten, P.: 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, pp. 166–178. ACM, New York (2008)