Technical Report 12-07: Understanding Lack of Trust in Distributed Agile Teams: A Grounded Theory Study

Siva Dorairaj, James Noble and Petra Malik

February 21, 2012

Abstract

Background: Trust fuels team performance and contributes to build an effective and cohesive team. The self-organizing and collaborative nature of Agile teams increases the importance of trust in software development teams. Trust is, however, affected in distributed teams. Aim: Through a Grounded Theory study we investigate the impact of trust in distributed Agile teams. Method: We interviewed 45 Agile practitioners from 28 different software companies in the USA, India and Australia, over a period of 3 years, using semi-structured open-ended questions. Results: In this paper, we present the reasons for lack of trust and its adverse effects in distributed Agile teams that emerged from the analysis, using the causal-consequences theoretical model. Conclusion: Understanding the causes and consequences of lack of trust can develop awareness of the importance of trust in distributed teams and pave ways for effectively building trust in project-oriented contexts.

1 Introduction

The success of Agile software development in delivering quality software on time and on budget lures software companies to use Agile methods in distributed software development, mainly to capitalise on the talented global resources pool and to lower costs [7, 55, 63]. Agile methods, however, advocate frequent face-to-face interaction and close collaboration between team members including barrier-free collocated teams [10, 12, 20]. There is no doubt that integrating Agile methods in distributed software development requires considerable effort from team members and managers [7, 41, 55, 58].

We conducted a Grounded Theory study to explore distributed software development from the perspective of Agile practitioners. This study, over a period of three years, involved 45 Agile practitioners from 28 different software companies in the USA, Australia and India. We conducted one-on-one face-to-face semi-structured interviews with Agile practitioners from a wide range of roles in distributed Agile projects. The initial analysis gave rise to a number of key concerns for distributed teams, particularly *cultural differences*, *communication*, *team interaction*, and *trust*. We formulated research questions for these emergent key concerns and commenced in-depth investigation on them.

In this paper we investigate the research question: *"What are the causes and consequences of lack of trust in distributed Agile teams?"*, whereas in a companion paper [19], we investigated the research question *"What are the techniques for building trust in distributed Agile teams?"* and presented seven techniques for building trust.

2 Background

Agile methods are a family of software development methods that follow an iterative and incremental style of development [1]. The Agile methods includes XP (eXtreme Programming) [6], Scrum [62], ASD (Adaptive Software Development) [36], DSDM (Dynamic Systems Development Method) [66], FDD (Feature Driven Development) [56], and Crystal Clear [11].

Scrum and XP are considered to be the most widely adopted Agile methods in software development projects [23]. Scrum practices such as *sprint planning*, *daily meetings*, *sprint reviews*, and *sprint retrospectives* require trust motivated individuals to communicate effectively and collaborate closely [49, 62]. One of the 12 principles behind the Agile Manifesto explicitly states that managers should trust the Agile teams to "get the job done" [24]. Agile teams are collaborative and self-organizing in nature [40, 64]. The self-organizing nature of Agile teams increases the importance of trust in software development teams [39, 48, 50]. A member of a team should trust that other members in the team are competent, knowledgeable and willing to collaborate effectively to deliver business values to customers [37, 48, 53]. Trust among team members is imperative for blending agility with distributed software development projects [53, 58]. Trust is, however, affected in distributed teams [9, 51, 58]. Several studies discuss the impact of trust in software development teams [51, 5, 54, 45, 60, 42]. There is, however, a paucity of studies into the impact of trust in distributed Agile software development [35, 49]. To address this gap, we investigate the reasons for lack of trust and its adverse effects in distributed 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. GT was originally developed by Barney G. Glaser and Anslem L. Strauss [32]. GT is successfully being used to study the social nature of Agile teams [25, 40, 47, 58, 69]. 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 [61], and the literature on distributed Agile software development generally and in particular the impact of trust on distributed teams is still scarce [35, 49]. Secondly, GT allows researchers to study social interactions and the behaviour of people [32], and Agile methods focus on people and their interactions in software development teams [62]. Using Glaser's guidelines, the study started with a general area of interest – distributed Agile software development – rather than beginning with a specific research question that can lead to preconceived ideas or hypotheses of the research phenomenon [2, 29, 34, 38, 52]. The problems and its key concerns, however, emerged in the initial stages of data analysis [28, 29].

3.1 Context

This study gradually 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 experience in Agile projects, and (2) direct involvement in distributed Agile projects 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, the project durations varied from 6 to 24 months, and the team size varied from 8 to 50 people depending on the complexity of the projects.

In order to get a rounded perspective, we included participants from a range of different roles within the projects: Scrum Master, Agile Coach, Developer, Quality Analyst, Business Analyst, Product Owner, and Senior Management (e.g. Development Manager, Recruitment Director, and Director of Technology). 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

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 background of the project, challenges faced by distributed Agile teams, and the strategies adopted by teams to overcome those challenges. We phrased our questions carefully so that the concerns in distributed Agile software development would emerge from the participants rather than from our own agenda.

Data collection and analysis occurs simultaneously in a GT study. We particularly avoided collecting all the data during a specific data collection phase, and then analysing them in a subsequent data analysis phase. Rather, we started analysis during an interview, and after each interview. This process of data collection is called *theoretical sampling* [27, p.36]:

> "*Theoretical sampling is the process of data collection for generting theory whereby the analyst jointly collects, codes, and analyzes his data and decides what data to collect next and where to find them, in order to develop his theory as it emerges.*"

Several key concerns emerged from the initial analysis: *cultural differences*, *communication*, *team interaction*, *trust*, *knowledge sharing* and *management support*. We adjusted our interview questions, particularly in subsequent interviews, to focus on these key concerns. That is, an ongoing analysis guided the future interviews, and also the choice of future participants. The research questions were formulated based on the emergent key concerns, such as:

- *"What are the techniques for bridging cultural differences in distributed Agile teams?"*
- *"What are the causes of communication challenges in distributed Agile teams?", and "How do distributed teams overcome communication challenges?"*
- *"How do Agile teams promote team interaction in distributed software development?"*

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

We investigated these research questions and presented the findings in different papers [15, 16, 17, 18, 19]. In this paper, we investigate the research question described in section 1, which is *"What are the causes and consequences of lack of trust in distributed Agile teams?"*

3.3 Data Analysis

Interviews were transcribed and the transcripts were analyzed using open coding to explore the meaning in the data by searching for similarities and differences [3, 26]. Open coding breaks down, examines, compares, conceptualises and categorises the data [67]. We collated key points from the data and assigned a *code*, or a summary phrase, to each key point. For example, we analysed the interview transcript from participant *P1*. In Table 2, "K" indicates "key point", and suffix "P1" identifies participant P1 (e.g. key point 3 from participant P1 appears as $K_{P1}3$). Thus we can trace back, through the interview transcriptions, to the actual quote and context of each key point.

ID	Key Point	Code
$K_{P1}3$	Video conferencing captures the visual aspect in communication	communication tool
$K_{P1}6$	The Americans were very unaware of the Indian culture	cultural difference
$K_{P1}7$	Only some people can understand dialects	language barriers
$K_{P1}12$	The team is comfortable reporting to their own people who were with them	team ambassador
$K_{P1}19$	They didn't want us to hear them asking questions	lack of trust
$K_{P1}24$	I travelled to Chennai for a few weeks to help in training and team building	team building
$K_{P1}35$	The Indian team members haven't even been engaged on the calls	lack of team spirit

Table 2: Examples of Key Points and Codes - data from participant P1

Using GT's *constant comparison method* [31], 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. This *theoretical memoing* is the "core stage" of a GT study [29]. We sorted the collection of the theoretical memos because sorting "puts the fractured data back together" [27, p.116]. A grounded theory is fundamentally written from the theoretical sorting of memos, and therefore sorting the theoretical memos is an "essential step" in GT [27, p.116].

3.4 Theoretical Coding

The final step of data analysis is the theoretical coding that systematically generates *theoretical codes*. Theoretical coding integrates all the data, codes, concepts and categories into a set of seamless theoretical codes. Substantive codes are the emergent categories that describes the research phenomenon, whereas the theoretical codes are emergent abstractions that model the integration of substantive codes [27, 30]. Theoretical codes are emergent and effectively weave the 'fractured' substantive codes into an organized theory. "Substantive codes could be related without theoretical codes, but the result is usually confused, unclear theoretically, and/or typically connected by descriptive topics but going nowhere theoretically" [30, p.60].

Theoretical coding schemas known as *theoretical coding families* [27, 30] are readily available to assist researcher conceptualize how the categories and their properties may relate to each other. The sorted memos are weaved together in the theoretical coding families so that the explication of the research phenomenon exhibits a strong connections between the substantive codes [27, 28].

In this study, the emergent concepts *No Sense of Belonging*, *Sense of Vulnerability*, *Poor Team Bonding*, *Lack of Cultural Understanding*, *Missing Face-to-Face Interaction*, and *Ineffective Communication* describe the substantive code 'reasons for lack of trust'. Further, the emergent concepts *Lack of Commitment*, *Ineffective Collaboration*, *Team Conflict*, and *Poor Team Performance* describe the substantive code 'adverse effects of lack of trust'. Sensitivity to the different theoretical coding families that model the integration of substantive codes allowed the theoretical code *Lack of Trust* to emerge naturally from the emergent concepts and categories. By comparing our findings with the theoretical coding families, the *causal-consequence theoretical family* [27, 30] emerged to be the best 'fit' for the theoretical code *Lack of Trust*. Figure 1 illustrates the causal-consequences theoretical family for the *Lack of Trust*. This causal-consequence theoretical family explicates the *Lack of Trust* in terms of the (1) causes of lack of trust, and (2) its adverse consequences. Through integration of the sorted memos into the causal-consequence theoretical family, we realize that the findings appear "more plausible, more relevant and more enhanced".

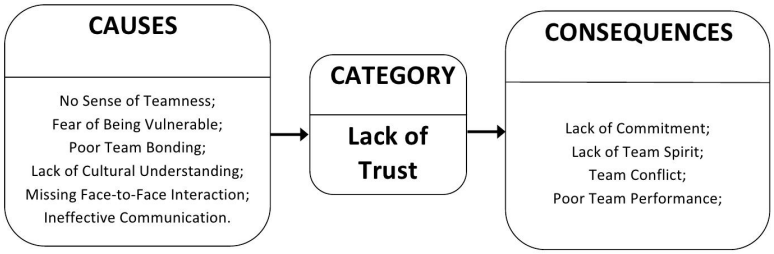

Figure 1: Causal-Consequences Theoretical Family for the *Lack of Trust*.

4 Causes of Lack of Trust

In this section, we describe the causes of lack of trust in distributed Agile teams based on the emergent concepts *No Sense of Belonging*, *Sense of Vulnerability*, *Poor Team Bonding*, *Lack of Cultural Understanding*, *Missing Face-to-Face Interaction*, and *Ineffective Communication*. We also present selected quotations drawn from our interviews that describe these concepts.

4.1 No Sense of Belonging

An individual's sense of belonging to a particular team is central to the success of the team because it provides the 'glue' that can promote desirable team cohesion for individuals working together from different locations, and foster trust building in the team [4, 21]. Without seeing and knowing the entire team, however, individuals in a team feel 'not part of the team':

"*I have not seen the entire team. So it's difficult to have a feel about the team members and get to trust them.*" – P24, Agile Coach.

Distributed team members often communicate over technology-mediated communication such as emails, telephones and video-conferences [22, 46]. Unlike video communication, the written communication and voice communication limits team interaction. Building rapport between development team and customers, and establishing trust in the team can be difficult when team interaction is limited:

"*All the team members have not physically met. We have only had conversations over the phone, and we don't do video conferencing. We've not really built the rapport with this customer.*" – P25, Scrum Master.

When individuals feel a strong identification with the team, they are willing to devote time and efforts for the achievement of their project goals. Team cohesiveness that is important for building trust can be affected when team members do not know each other:

"*We used to work together daily, but we did not know each other. I did not have a face for a name. It is hard to get the feeling of a team when you don't know who you are working with.*" – P9, Developer.

Team members may not have met the entire team due to the cost involved in getting team members travel to different locations, the inconveniences to organise the travel, and that management do not perceive that physical meetings are necessary to perform the job effectively. When team member do not know each other well, building a sense of belonging to the team can be a challenge. Without a 'true' sense of belonging to a team, building trust can be difficult.

4.2 Sense of Vulnerability

A feeling of insecurity can develops within team members who are new to a project team. Such members can be reluctant to be transparent and open with other members about their weaknesses, and would feel afraid to admit the truth into what is really happening in the team:

"*When people are new to the teams, they're not very confident to talk openly about the problems. They would say, "Things are fine, everything is good". But actually things were not [fine].* " – P45, Senior Management.

Some development team members, particularly junior members, feel scared to make estimations of story points, velocity, or product backlog because they presume that other members of the team would perceive them as incompetent for making 'wrong estimations':

"*The development team members feel scared to make estimations because they think that they'll be questioned if they have made wrong estimates.*" – P25, Scrum Master.

Team members avoid situations that expose their ignorance to the customers, particularly on the areas that they are expected to be the experts. They fear that the results of their 'inappropriate' decisions can be visible to the customers, and leave them feeling intimidated. A common way to avoid such a situation is not to engage with the customers:

"*The team were not engaged with the customer. The reason is always fear. Usually it's the fear of blame because they are often put in a situation where they have to make a decision, and the results of the decision will be very visible.* " – P32, Senior Management.

When a team member develops fear for admitting mistakes and weaknesses, it can be difficult to build trust with the team. Individuals should recognise that other members of the team do not have ill-intentions on one another, and therefore no need to be self-protective around the team. The feeling of insecurity and fear of blames make it impossible to build trust in the team.

4.3 Poor Team Bonding

Team bonding is about team relationships that fosters a sense of belonging and a feeling of togetherness within the team. Bonding brings the team together, stimulates team spirit and fosters the team to perform as a whole [63, 68]. When team members do not work closely together, bonding can be difficult, and building trust can be affected:

"*When I don't work with them on an everyday basis, I don't really know them. It doesn't always work for me to speak directly to them. I should either build that trust, or I need to find somebody to do that for me.*" – P44, Product Owner.

Poor bonding between team members can exacerbate the frequent conflicting ideas and opinions that arise in distributed teams that often consists of people from diverse backgrounds:

"*We have conference calls and all but people are just names, someone you see on the screen. You don't get to bond with them, and you don't understand them or their decisions.*" – P12, Developer.

Often distributed team members participate in technology-mediated distance training activities because it is costly and impractical for geographically distributed team members to meet face-to-face for group trainings. Coaches find it challenging to conduct effective trainings for team members whom they do not know personally:

"*Coaching someone that we have never seen his face, and never have personal connection is very difficult [because] you need to keep a trust motivated relationship with people you are coaching.* " – P7, Agile Coach.

Missing face-to-face interaction and continuous communication between distributed team members results in a poor team bonding. Further, team bonding activities to promote a feeling of togetherness are difficult for distributed teams. When members of a team do not have a strong team bonding, building trust among them can be difficult.

4.4 Lack of Cultural Understanding

Team members from different cultural backgrounds often face difficulties to understand each other's cultures [70]. A lack of cultural understanding can affect the team's effort for building trust and other aspects of the team's work:

"*The project manager in the USA had great difficulty initially to understand our culture and the way we do things. She was not comfortable with us and did not much trust us.*" – P45, Senior Management.

Some spoken words that have different meanings in different cultures can give rise to unnecessary conflicts between individuals, and create circumstances to loose trust among those particular individuals:

"One of the experiences is when asked a question [to the Indian team], the answer is always "Yes!" even though they didn't know it. We later came to understand that "Yes" means "Yes, I heard you". But, here [in the USA] we always assume that "Yes!" means "Yes, it's done". – P1, Developer.

Participants discuss that culture affects trusted relationships in the distributed teams, and engendering cultural awareness among individuals of different cultures promotes an understanding among them to focus on the similarities rather than the differences:

"*One big thing required for distributed teams is cultural awareness. Culture definitely affects the trust and team building. Get people to understand the [cultural] differences, and get beyond the differences.*" – P31, Agile Coach.

The cultural differences include the accent and rapidness of verbal communication, body language such as head movements or hand gestures, and the actual meaning for the spoken words. The lack of cultural understanding in a team contributes significantly to the lack of trust and poor team bonding among members of the teams.

4.5 Missing Face-to-Face Interaction

Face-to-face interaction provides opportunity to socialize with team members, and build or sustain trust and team relationships [14]. Participants discuss that the missing of continuous face-to-face interaction affects trust building in distributed teams:

"*I think it is really hard to maintain trust, especially harder to establish trust without having a face-to-face interaction. If trust is not there, it is hard to succeed as a team.*" – P4, Agile Coach.

Individuals rely on multiple modes of communication in face-to-face interaction, such as paraverbal (e.g. voice volume, tone of voice) and non-verbal (e.g. facial expression, hand gestures) cues. Through technology-mediated communication, lack of these cues reduces the richness of the information exchanged between the team members, and subsequently affects trust in the team:

"*Lack of face-to-face meetings reduces the visibility and transparency into the status of the project, which leads to a mood of mistrust between team members in different locations..*" – P41, Agile Coach.

When team members in different locations interact through technology-mediated communication, it can be difficult to gauge their level of interest and comprehension on the discussion. Some team members may not trust that other members are attentive and fully engaged in the conversation:

> "*With distributed team, you're on one side and you're telling the team on the other side what to do. And, you're hoping that they are getting it, you're hoping that they are not surfing the net while listening to you. If they've switched off, you don't even know it!*" – P44, Product Owner.

Opportunities to meet face-to-face for distributed teams, though rare, can happen during project inceptions and when team members get to travel to a different location. Continuous face-to-face interaction is, however, difficult for most team members. This missing of face-to-face interaction, makes it much difficult to build trust in distributed teams.

4.6 Ineffective Communication

Scrum meetings provide avenues for members of a team to communicate with the entire team on a daily basis, and report the impediments and update the progress of the work.
Ineffective communication among team members occurs when individuals misunderstood the information that was exchanged, non-verbal communication such as hand gestures and facial expressions were not recognised, or attention to subtle information such as emotion of an individual was not noticed [8]. Ineffective communication, specifically during the Scrum meetings, can affect trust and social bonding among team members:

> "*Somebody will go on sharing for a long time without much focus, or somebody will not properly articulate their impediments. Even somebody will come really late to the stand-up meetings. In the beginning, it was difficult to build trust.*" – P39, Agile Coach.

Some team members who were collaborating remotely did not leverage efficiently the communication technologies such as video, audio, or written that were available for them. Not using the right technology hinders effective communication in the team:

> "*We mostly used written communication – chats and a lot of emails. We should have used video or voice instead of written communication.* " – P9, Developer.

Management in some organizations limits the availability of communication tools and technologies for the project team:

> "*We need web cameras but it's against our organisational policy. So we don't use web cameras in our distributed projects.*" – P24, Agile Coach.

Without suitable tools and technologies, team members in different locations can encounter difficulties to communicate effectively. Ineffective communication affects trust building, team interaction, and team bonding, particularly between members from different cultures in different geographical locations.

5 Consequences of Lack of Trust

In this section, we describe the consequences of lack of trust in distributed Agile software development based on the concepts *Lack of Commitment*, *Ineffective Collaboration*, *Team Conflict*, and *Poor Team Performance*.

5.1 Lack of Commitment

Commitment to a team often translates into a willingness to collaborate with team members and improved team performance [13]. Conversely, a lack of commitment reduces team performance and leaves the team members not willing to participate in the project activities. Lack of commitment arises when team members were dissatisfied with the outcome of a discussion, or team members feel that they have not been heard when a decision is made. Hence, they do not trust that the decision was effective and do not buy-in to the decision. Lack of commitment even from just one member in the team can affect the overall performance of the team, and reduce team velocity and productivity:

> "*Lots of times we missed release deadlines. It was not that we couldn't find the solutions but because of [lack of] trust, the team was not committed.* " – P8, Developer.

When team members feel that they have not been given adequate opportunity to express themselves, they become rather passive in the discussion and do not wish to contribute to the ongoing discussions:

> "*Meetings were happening without any purpose, without any useful discussions, and without any active participation from the team.*" – P43, Agile Coach.

In some worst cases, team members do not show interest in delivering basic business values to the customer. This can affect not only the ongoing project, but also future projects from that particular customer:

> "*The team never prepared properly for the demos, and [the demos] weren't 'business meaningful' to anyone. There was just a lack of trust amongst the team members. People should be honest about what's preventing them from moving forward.*" – P30, Agile Coach.

Lack of commitment can force team members to disregard their responsibilities and reduce collaboration with the rest of the team members. Teams should ideally move forward with complete buy-in from every member, even those who suggested alternative ideas or disagreed to the ideas that other members suggested.

5.2 Ineffective Collaboration

Trust motivated individuals in a cohesive team are willing to collaborate and cooperate with others in the team. Collaboration brings together the knowledge, experience and skills of team members to achieve the collective goals of a project [4]. Conversely, in the absence of trust, collaboration was affected in the team. Some customers and managers only collaborated with some selected members rather than the entire team:

"*When a controversial decision has to be made, the client and [project] manager would listen to developers there [in the USA] than the developers in India.*" – P12, Developer.

Some team members could not collaborate with other members in the team because one side of the team did not engaged them in the activities, such as a decision making, that ideally require all the team members to participate:

"*In a distributed [project], a decision has to be made by the teams in both locations. But, we haven't the ability to make our own decision. Mostly the Indian team will be waiting for a decision from the USA team. Trust in team is crucial [for collaboration].*" – P29, Scrum Master.

Some team members were only willing to complete the bare essentials of their job, and were reluctant to collaborate further to assist other team members collectively achieve the project goals:

"*It is hard to convince the junior developers that their work is not done after writing the code alone.*" – P25, Scrum Master.

Agile methods particularly emphasize on collaboration. The Agile Manifesto states that individuals and their interactions, and customer collaboration are valuable for Agile projects [24]. One of the principles behind the Agile Manifesto states that "business people and developers *must* work together daily throughout the project" [24]. The successful realization of Agile software development projects is closely associated with the effective collaboration between the team members, and therefore it is imperative for team members to trust each other in order to 'work well together'.

5.3 Team Conflict

Conflict are serious disagreements that arises from the clash of opinions, perceptions, or values in the area where individuals are concerned about the outcome [33, 65]. A team member recognises the differences in approaches to solving a problem and perceives that one's approach is correct and the best. This results in conflicts that end in discussions where the team members openly discuss their own point of view and argue against other's point of views. Such 'healthy and constructive' conflicts are in fact needed to produce effective and meaningful decisions.

Participants, however, describe that conflicts in the distributed teams often arise from the absence of trust among team members. Such conflicts can stir emotions of an individual, and give rise to negative behaviours such as 'finger-pointing' and 'backstabbing' between the members:

"*They are being political because they seem to be your friend but they are stabbing you at the back. There's a big lack of trust. I find it very hard to work with the rest of the team.*" – P6, Developer.

Customers whom did not trust the development team members have unrealistic expectations that hurt the team. Unresolved conflicts can distract team members, undermine team spirit and, ultimately, affect team performance:

"*The customer's expectation is that we are available at any point of the day and night. Maybe when we have the customer's trust, we'll be in a position to say, "Hey customer, we will not be available in the middle of the night. This is hurting the team".* " – P25, Scrum Master.

Some team members do not return phone calls or e-mails, hoards information that need to be shared, and do not follow-up to resolve the problems that arise in the projects. Failing to confront conflicts that arise in the team can worsen the conflict and effect customer satisfaction:

"*We let the customers know about this [problem] seven months earlier but we didn't [follow-up] with them. When we install the software, the problem started appearing and the customers were shouting at us that it didn't work. Basically, there was no trust in the team.*" – P9, Developer.

Conflicts are inevitable in a diverse team that comprises of knowledgeable individuals working in the area of their expertise. In the absence of trust, when conflicts arise, some team members start blaming someone for it, or completely ignore the situation that causes the conflict rather than resolving the problem. A conflicts has to be acknowledged, managed and resolved so that it does not negatively affect the customer satisfaction and team performance.

5.4 Poor Team Performance

A standard level of performance is expected from the members of a team on the initiative and effort that the team members are able to demonstrate [4, 43]. An Agile team that follows the principle to 'satisfy the customer through early and continous delivery of valuable software' [24], however, expects a significantly high level of performance from its members. Poor performance of a team means that the team has not achieved the outcomes that were collectively agreed between the development team and the customer:

"*When trust breaks down, the ability of the team is affected, the velocity of the project goes down, [and] you will not get that linear [and] stable velocity across team members.*" – P3, Scrum Master.

Some team members were delaying their work, and some did not put in enough effort to achieve the goals of the project:

"*Here some team members have the tendency to hide the problems or delay the work. The team is not focused on its goal.* " – P30, Agile Coach.

In some cases, poor team performance can lead to failure of a software development project. Teams that did not collaborate and communicate effectively, and build trust in the team were not able to meet the goals of the project:

"*We were behind [schedule], not on the right track, and the project did not meet the client's expectation. At this level, we had to build the trust that was lacking in the first six months of the project, and request the client to allow us to do [the project] all over again.*" – P19, Developer.

Lack of trust affects team performance in a varying degree. Some team members did not monitor and provide feedback on each others performance, whereas some were not focused on the tasks, deadlines and deliverables. Poor performance of even one member of a team can affect the overall performance of the entire team.

6 Discussion and Related Work

In this section we discuss our findings in the light of related work. We realize that trust plays an important role in determining the success and failure of both Agile and non-Agile projects for collocated and distributed teams. The knowledge on what causes trust to decline, what are the adverse effects of absence of trust and how to build trust in a team, can create an awareness on the importance of trust in software development teams. Several studies particularly investigate the impact of trust in non-Agile teams [5, 45, 51, 54, 60], and Agile teams [35, 48]. We are only able to discuss selected studies due to space constaints.

On investigating mature XP teams, Robinson and Sharp [59] argue that absence of trust would affect the sense of respect, responsibility, concern for the quality of working life and faith in the ability of the Agile team, and members of the team would doubt that the team as a whole could deliver business values to customers. We found that, in the absence of trust, team members exhibited lack of commitment to the work being carried out. Some members were not engaged in useful discussions, some members were not effectively participating in the meetings, and some members were not prepared for the demos with the customers. The team members disregarded their responsibilities to the customer and fellow members of the team in one way or another.

Moe and Smite [51] conducted an empirical study to understand the causes and consequences of lacking trust in global software development in four software projects. All these projects report that lack of trust affected team performance and resulted in a decrease in product quality and team velocity. From our study we found that, in the absence of trust, team members were less focused on the project goals, the projects were behind schedule, and the project deliverables did not meet customer's expectation.

Based on a study on customer communication in a large globally distributed Agile software development, Korkala, Pikkarainen and Conboy [44] found that the lack of trust was one of the reasons for the customer not involved in the implementation of distributed Agile projects. The reseachers suggest that efficient communication is one of the most essential factors in distributed software development. Piccoli and Ives [57] report the findings from a longitudinal study of virtual teams that incongruence, particularly obstacles to effective communication, and reneging create the potential for trust decline. We found that ineffective communication, particularly during the daily meetings and demo for customers, and failure to recognize the need for different communication tools and techniques for the team members, create circumstances for trust decline.

Through a case study of an outsourced information systems development project, Lander, Purvis, McCray and Leigh [45] argue that building trust amongst team members in different locations is especially difficult because the individuals involved in the project often have little or no prior experiences working together with other individuals in the team, and yet rely on one another's expertise and judgement for a successful project. Oza, Hall, Rainer and Grey [54] conducted a study based on an empirical investigation of eighteen mature software companies located in India.

The reseachers describe several critical success factors to achieving an initial trust, and eventually maintaining trust in software outsource relationships, and suggest that trust is considered to be very fragile in outsourcing relationships. Our participants acknowledged that building trust between team members from different locations in project-oriented contexts can be difficult, and therefore some distributed teams take the necessary measures to avoid the causes for trust to decline in the team.

7 Limitation

The inherent limitation of a Grounded Theory study is that the findings are grounded in the specific contexts explored in the research [2, 38]. 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.

8 Conclusions

Software companies are increasingly venturing into distributed Agile software development. Several concerns need to be addressed to realise the benefits of integrating Agile methods in distributed projects. Through a Grounded Theory study that was carried out over a period of three years and gradually involved 45 Agile practitioners, several key concerns emerged from our analysis. Some of them were common to any distributed software development, whereas some were specific for distributed Agile software development. It was found that trust is an important concern generally for Agile teams, and particularly for distributed Agile teams. The self-organizing nature of the 'empowered' Agile teams, that typically hold significant authority to perform highly interdependent tasks, increases the importance of trust in software development. Hence, we formulate related research questions to investigate the impact of trust in distributed teams. In this paper we report the causes and consequences of lack of trust, and in a companion paper [19] we report the techniques for building trust in distributed Agile teams. There is an increasing realisation that understanding the causes and consequences of lack of trust in teams can pave ways for building trust in project-oriented contexts, and subsequently contribute to the success of a distributed Agile project.

We acknowledge that there can be other causes and consequences of lack of trust that can be useful and effective in their own contexts, but did not emerge from our analysis. The understanding of trust in distributed Agile projects gained from this study could be used as a foundation for conducting future studies involving teams working on different Agile projects in different contexts. Further studies could also compare successful distributed Agile projects with unsuccessful projects to consider the ramifications of trust.

9 ACKNOWLEDGMENTS

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

[4] G. Asproni. Motivation, teamwork, and Agile development. *Agile Times*, 4(1):8–15, 2004.

[5] M. A. Babar, J. M. Verner, and P. T. Nguyen. Establishing and maintaining trust in software outsourcing relationships: An empirical investigation. *Journal of Systems and Software*, 80(9):1438–1449, 2007.

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

[7] I. Bose. Lessons learned from distributed Agile software projects: A case-based analysis. *Communications of the Association for Information Systems*, 23(1):619–632, 2008.

[8] R. P. Bostrom. Successful application of communication techniques to improve the systems development process. *Information & Management*, 16(5):279–295, 1989.

[9] E. Carmel. Thirteen assertions for globally dispersed software development research. In *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, volume 3, pages 445–452, 1997.

[10] A. Cockburn. *Agile Software Development*. Addison-Wesley, Indianapolis, 2002.

[11] A. Cockburn. *Crystal Clear: A Human-Powered Methodology for Small Teams*. Addison-Wesley Professional, 2004.

[12] A. Cockburn and J. Highsmith. Agile software development: The people factor. *Computer*, 34(11):131–133, 2001.

[13] A. Crossman and L. Lee-Kelley. Trust, commitment and team working: The paradox of virtual organizations. *Global Networks*, 4(4):375–390, 2004.

[14] K. Crowston, J. Howison, C. Masango, and U. Y. Eseryel. The role of face-to-face meetings in technology-supported self-organizing distributed teams. *Professional Communication, IEEE Transactions on*, 50(3):185–203, Sept. 2007.

[15] S. Dorairaj, J. Noble, and P. Malik. Understanding the importance of trust in distributed Agile projects: A practical perspective. In *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.

[16] 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.

[17] S. Dorairaj, J. Noble, and P. Malik. Effective communication in distributed Agile software development teams. In *Agile Processes in Software Engineering and Extreme Programming*, volume 77 of *Lecture Notes in Business Information Processing*, pages 102–116. Springer Berlin Heidelberg, 2011.

[18] 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.

[19] 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.

[20] T. Dybå and T. Dingsøyr. What do we know about Agile software development? *IEEE Software*, 26(5):6–9, Sept/Oct 2009.

[21] C. M. Fiol and E. J. O'Connor. Identification in face-to-face, hybrid, and pure virtual teams: Untangling the contradictions. *Organization Science*, 16(1):19–32, 2005.

[22] S. M. Fiore. Distributed coordination space: Toward a theory of distributed team process and performance. *Theoretical Issues in Ergonomics Science*, 4(3–4):340–364, 2003.

[23] B. Fitzgerald, G. Hartnett, and K. Conboy. Customising Agile methods to software practices at Intel Shannon. *European Journal of Information System*, 15(2):200–213, 2006.

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

[25] 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.

[26] 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.

[27] B. Glaser. *Theoretical Sensitivity: Advances in Methodology of Grounded Theory.* Sociology Press, Mill Valley, CA, 1978.

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

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

[30] B. Glaser. *The Grounded Theory Perspective III: Theoretical Coding.* Sociology Press, Mill Valley, CA, 2005.

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

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

[33] D. H. Gobeli, H. F. Koenig, and I. Bechinger. Managing conflict in software development teams: A multilevel analysis. *Journal of Product Innovation Management*, 15(5):423–435, 1998.

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

[35] E. Hasnain and T. Hall. Investigating the role of trust in Agile methods using a light weight systematic literature review. In *Agile Processes in Software Engineering and Extreme Programming*, volume 9 of *Lecture Notes in Business Information Processing*, pages 204–207. Springer Berlin Heidelberg, 2008.

[36] J. A. Highsmith,III. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems.* Dorset House Publishing, New York, USA, 2000.

[37] R. Hoda. *Self-Organising Agile Teams: A Grounded Theory.* PhD thesis, Victoria University of Wellington, New Zealand, 2011.

[38] 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.

[39] R. Hoda, J. Noble, and S. Marshall. Balancing acts: Walking the Agile tightrope. In *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, pages 5–12, New York, NY, USA, 2010. ACM.

[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] E. Hossain, M. A. Babar, and J. Verner. Towards a framework for using Agile approaches in global software development. In *Product-Focused Software Process Improvement*, volume 32 of *Lecture Notes in Business Information Processing*, pages 126–140. Springer Berlin Heidelberg, 2009.

[42] S. L. Jarvenpaa, T. R. Shaw, and D. Staples. Toward contextualized theories of trust: The role of trust in global virtual teams. *Information Systems Research*, 15, 2004.

[43] M. C. Jones and A. W. Harrison. Is project team performance: An empirical assessment. *Information & Management*, 31(2):57–65, 1996.

[44] M. Korkala, M. Pikkarainen, and K. Conboy. Distributed Agile development: A case study of customer communication challenges. In *Agile Processes in Software Engineering and Extreme Programming*, volume 31 of *Lecture Notes in Business Information Processing*, pages 161–167. Springer Berlin Heidelberg, 2009.

[45] M. C. Lander, R. L. Purvis, G. E. McCray, and W. Leigh. Trust-building mechanisms utilized in outsourced IS development projects: A case study. *Information & Management*, 41(4):509–528, 2004.

[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 team: A grounded theory. In *Proceedings of the AGILE*, pages 57–64, 2009.

[48] O. McHugh, K. Conboy, and M. Lang. The impact of Agile practices on trust in software project teams. *Software, IEEE*, PP(99):1, 2011.

[49] O. McHugh, K. Conboy, and M. Lang. Using Agile practices to build trust in an Agile team: A case study. In *Information Systems Development*, pages 503–516. Springer New York, 2011.

[50] N. B. Moe, T. Dingsoyr, and T. Dyba. Understanding self-organizing teams in Agile software development. In *ASWEC 2008, 19th Australian Conference on Software Engineering*, pages 76–85, March 2008.

[51] N. B. Moe and D. Šmite. Understanding lacking trust in global software teams: A multi-case study. In J. Münch and P. Abrahamsson, editors, *Product-Focused Software Process Improvement*, volume 4589 of *Lecture Notes in Computer Science*, pages 20–34. Springer Berlin Heidelberg, 2007.

[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] N. V. Oza, T. Hall, A. Rainer, and S. Grey. Trust in software outsourcing relationships: An empirical investigation of Indian software companies. *Information and Software Technology*, 48(5):345–354, 2006.

[55] M. Paasivaara and C. Lassenius. Could global software development benefit from Agile methods? In *Proceedings of the IEEE International Conference on Global Software Engineering*, pages 109–113, Washington, DC, USA, 2006. IEEE Computer Society.

[56] S. Palmer and M. Felsing. *A Practical Guide to Feature-Driven Development*. Pearson Education, 2001.

[57] G. Piccoli and B. Ives. Trust and the unintended effects of behavior control in virtual teams. *MIS Quarterly*, 27(3):365–395, 2003.

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

[59] H. Robinson and H. Sharp. The characteristics of XP teams. In J. Eckstein and H. Baumeister, editors, *Extreme Programming and Agile Processes in Software Engineering*, volume 3092 of *Lecture Notes in Computer Science*, pages 139–147. Springer Berlin / Heidelberg, 2004.

[60] R. Sabherwal. The role of trust in outsourced IS development projects. *Communications of ACM*, 42:80–86, February 1999.

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

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

[63] L. Seiyoung and Y. Hwan-Seung. Distributed Agile: Project management in a global environment. *Empirical Software Engineering*, 15(2):204–217, 2010.

[64] H. Sharp and H. Robinson. Collaboration and coordination in mature eXtreme programming teams. *International Journal of Human-Computer Studies*, 66(7):506–518, 2008. Collaborative and Social Aspects of Software Development.

[65] K. K. Smith and D. N. Berg. A paradoxical conception of group dynamics. *Human Relations*, 40(10):633–657, October 1987.

[66] J. Stapleton. *Dynamic Systems Development Method: The Method in Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

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

[68] S. D. Teasley, L. A. Covi, M. Krishnan, and J. S. Olson. Rapid software development through team collocation. *Software Engineering, IEEE Transactions on*, 28(7):671–683, July 2002.

[69] 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.

[70] J. Winkler, J. Dibbern, and A. Heinzl. The impact of cultural differences in offshore outsourcingcase study results from GermanIndian application development projects. *Information Systems Frontiers*, 10:243–258, 2008.