Information and Software Technology 53 (2011) 521–534

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

The impact of inadequate customer collaboration on self-organizing Agile teams

Rashina Hoda *, James Noble, Stuart Marshall

School of Engineering and Computer Science, Victoria University of Wellington, New Zealand

ARTICLE INFO

Article history:
Available online 16 November 2010

Keywords:
Agile software development
Customer collaboration
Agile Undercover
Grounded Theory

ABSTRACT

Context: Customer collaboration is a vital feature of Agile software development.
Objective: This article addresses the importance of adequate customer involvement on Agile projects, and the impact of different levels of customer involvement on real-life Agile projects.
Method: We conducted a Grounded Theory study involving 30 Agile practitioners from 16 software development organizations in New Zealand and India, over a period of 3 years.
Results: We discovered that *Lack of Customer Involvement* was one of the biggest challenges faced by Agile teams. Customers were not as involved on these Agile projects as Agile methods demand. We describe the causes of inadequate customer collaboration, its adverse consequences on self-organizing Agile teams, and *Agile Undercover* — a set of strategies used by the teams to practice Agile despite insufficient or ineffective customer involvement.
Conclusion: Customer involvement is important on Agile projects. Inadequate customer involvement causes adverse problems for Agile teams. The *Agile Undercover* strategies we've identified can assist Agile teams facing similar lack of customer involvement.

1. Introduction

Customer collaboration in traditional software development projects is typically limited to providing the requirements in the beginning and feedback towards the end, with limited regular interactions between the customer and the development team [1–4]. In contrast, customer collaboration is a vital feature and an important success factor in Agile software development [5–10]. Agile methods expand the customer role within the entire development process by involving them in writing user stories, discussing product features, prioritizing the feature lists, and providing rapid feedback to the development team on a regular basis [1,4,9,11]. But are customers on real-life Agile projects involved in their projects to the extent Agile methods demand? What are the causes of lack of customer involvement if indeed the customers are not adequately involved? What are the consequences of inadequate customer involvement on the self-organizing Agile teams? And finally, can Agile teams continue to practice Agile when customers do not collaborate adequately?

We found answers to all of these questions through a Grounded Theory study involving 30 Agile practitioners from 16 different software development organizations in New Zealand and India. Over the period of the research spanning 3 years, we discovered that *Lack of Customer Involvement* was one of the biggest concerns of the majority of the participants. As a part of the study, we conducted semi-structured, face-to-face interviews with Agile practitioners in the New Zealand and Indian software industries, using open-ended questions. We also supplemented this data with observations. We analyzed the data using Grounded Theory's *Constant Comparison* method and found that *Lack of Customer Involvement* emerged as one of the most common themes from the analysis.

The participants identified several causes for lack of customer involvement, including: skepticism and hype, the distance factor, lack of time commitment, dealing with large customers, fixed-bid contracts, and ineffective customer representatives. Inadequate customer involvement on these Agile projects led to adverse consequences for the self-organizing Agile teams, including: pressure to over-commit, problems in gathering and clarifying requirements, problems in prioritizing requirements, problems in securing feedback, loss of productivity, and in extreme cases, business loss. The practitioners had developed some strategies to overcome the lack of customer involvement on these projects and continue to practice Agile despite insufficient or ineffective customer involvement. We name these strategies *Agile Undercover*. In this article, we describe each of these causes, consequences, and *Agile Undercover* strategies in detail.

Our contributions in this article are: (a) a description of the causes of lack of customer involvement on these Agile projects, (b) its adverse consequences faced by self-organizing Agile teams, (c) *Agile Undercover* strategies used by these Agile teams to practice

* Corresponding author. Tel.: +64 463 5233x8486; fax: +64 463 5045.
E-mail addresses: rashina@ecs.vuw.ac.nz (R. Hoda), kjx@ecs.vuw.ac.nz (J. Noble), stuart@ecs.vuw.ac.nz (S. Marshall).
URLs: http://www.ecs.vuw.ac.nz (R. Hoda), http://www.ecs.vuw.ac.nz (J. Noble), http://www.ecs.vuw.ac.nz (S. Marshall).

0950-5849/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2010.10.009

Agile despite insufficient or ineffective customer involvement, (d) description of our application of Grounded Theory, including the six C's theoretical model to explain the causes, consequences, and strategies, and (e) a discussion of the implications of our results for theory and practice.

This journal article is an extension of our conference paper on the same topic [5]. In particular, this article expands and extends the conference paper in the following ways: it provides a detailed description of the research methodology, expands on the results by including other causes, consequences, and *Agile Undercover* strategies not covered in the conference paper due to space constraints, discusses this work in context of related research, and explicates the implications for theory, for practice, for future work, and limitations of the study.

The rest of the article is structured as follows: Section 2 presents a brief 'Background' on Agile software development and the level of customer collaboration recommended by Agile methods in theory. Section 3 describes our 'Research method' followed by the 'Results' of the study in Section 4. The next section, Discussion, presents the related work, implications of our research findings, limitations of the study, and avenues for future work. The paper concludes in Section 6.

2. Background

Agile software development methods emerged in the late 1990s [12]. The term 'Agile' was adopted as the umbrella term for methods such as Scrum [13], XP (eXtreme Programming) [14], Crystal [15], FDD (Feature-Driven Development) [16], DSDM (Dynamic Software Development Method) [17], and Adaptive Software Development [18]. Scrum and XP are considered to be the most widely adopted Agile methods in the world [19]. XP focuses on developmental practices, while Scrum mainly covers project management [7].

Agile teams are self-organizing teams [6,20,8,21–23] composed of "*individuals [that] manage their own workload, shift work among themselves based on need and best fit, and participate in team decision making.*" [24]. Takeuchi and Nonaka [25] describe self-organizing teams as exhibiting autonomy, crossfertilization, and self-transcendence. Self-organizing teams must have common focus, mutual trust, respect, and the ability to organize repeatedly to meet new challenges [20]. Self-organizing teams are not leaderless, uncontrolled teams [20,25]. Leadership in self-organizing teams is meant to be light-touch and adaptive [26], providing feedback and subtle direction [27,28,25]. Leaders of Agile teams are responsible for setting direction, aligning people, obtaining resources, and motivating the teams [27]. Agile projects have job titles such as Scrum Masters [13] and (XP) Coaches [29] instead of traditional managers.

Agile methods advocate high levels of collaboration between the team and their customers in order to frequently release product features that deliver business value in each iteration. Several Agile practices demand customer collaboration such as planning, prioritizing, reviewing, and providing feedback. In Scrum, the customer is an important part of the process [13]. The customer representative, also known as Product Owner, is responsible for defining the features of the product; prioritizing the list of features (backlog); reviewing the developed features; and providing feedback to the team. The development team is meant to use the backlog to develop corresponding features every iteration in close collaboration with the customer [13]. Similarly, XP, has a established customer role [14] that is responsible for providing and prioritizing requirements; assessing the developed tasks against a set of customer-defined acceptance criteria; and providing frequent feedback to the team [14].

We have presented a brief background of Agile software development and the level of customer collaboration recommended by Agile methods. We discuss related works in greater detail in Section 5, after we present our research results such that the order of presentation reflects the order in which the literature review was carried out as per classic Grounded Theory guidelines.

3. Research method

Grounded Theory (GT) is the systematic generation of theory from data analyzed by a rigorous research method [30,31]. GT was developed by sociologists Glaser and Strauss [32]. We chose GT as our research method for several reasons. Firstly, Agile methods focus on people and interactions and GT, used as a qualitative research method, allows us to study social interactions and behaviour. Secondly, GT is most suited to areas of research which have not been explored in great detail before, and the research literature on Agile team-customer relationships is scarce [1]. Finally, GT is being increasingly used to study Agile teams [33,9,34,35].

The aim of GT is to generate a theory as an interrelated set of hypotheses generated through constant comparison of data at increasing levels of abstraction. In generating a theory, the GT researcher uncovers the main concern of the research participants and how they go about resolving it. The distinguishing feature of the GT method is the absence of a clear research problem or hypothesis upfront, rather the researcher tries to uncover the research problem as the main concern of the participants in the process [36,37]. As such, following Glaser's guidelines, we started out with a general area of interest — Agile project management — rather than beginning with a specific research question [35].

3.1. Data collection

Data collection in GT is guided by a process called *Theoretical Sampling*, an ongoing process which helps decide what data to collect next based on the emerging theory:

> "*Theoretical Sampling is the process of data collection for generating 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.*" [30]

3.1.1. Recruiting participants

We approached Agile practitioners in New Zealand and India primarily through user groups such as Agile Professionals Network (APN, NZ) and Agile Software Community of India (ASCI, India). Conferences and other events focused on Agile software development also provided opportunity to invite participants.

Our initial participants belonged to relatively new Agile teams. Using theoretical sampling, we were able to discern gaps in our emerging theory which prompted us to study more mature teams towards later stages of our research. We were also able to see the need to include participants from different aspects of software development at different stages of our research guided by the emerging theory, such as Agile coach, developer, tester, business analyst, customer, and senior management. Data collection by theoretical sampling helped us develop our emerging theory by (a) adapting questions to focus on emerging concerns (b) choosing participants that were better placed to provide information on the emerging concerns.

3.1.2. Interviews and observations

We collected data by conducting face-to-face, semi-structured interviews with Agile practitioners using open-ended questions. The interviews were approximately an hour long and focused on

the participants' experiences of working with Agile methods, in particular the challenges faced in Agile projects and the strategies used to overcome them. We also observed several Agile practices such as daily stand-up meetings (co-located and distributed), release planning, iteration planning, and demonstrations.

The interviews were voice recorded, transcribed, and analyzed. Data collection and analysis were iterative so that constant comparison of data helped guide future interviews and the analysis of interviews and observations fed back into the emerging results.

3.2. Data analysis

The procedure of data analysis — called *coding* in GT — can begin as soon some data is collected. There are two types of coding — *Substantive coding* and *Theoretical coding* — that give rise to Substantive and Theoretical codes, respectively. The substantive codes are "*the categories and properties of the theory which emerges from and conceptually images the substantive area being researched* [31]. In contrast, theoretical codes "*implicitly conceptualize how the substantive codes will relate to each other as a modeled, interrelated, multivariate set of hypothesis in accounting for resolving the main concern*" [31]. In this section, we describe the coding mechanisms — *Open Coding* and *Selective Coding* — that lead to substantive codes.

3.2.1. Open coding

Open coding is the first step of data analysis. We used open coding to analyze the interview transcripts in detail [30,38]. We began by collating key points from each interview transcript [38]. Then we assigned a *code* — a phrase that summaries the key point in 2 or 3 words — to each key point [1]. The codes arising out of each interview were constantly compared against the codes from the same interview, and those from other interviews and observations. This is GT's *Constant Comparison* method [39,32] which was used again to group these codes to produce a higher level of abstraction, called *concepts* in GT.

The constant comparison method was repeated on the concepts to produce another level of abstraction called a *category*. As a result of this analysis, the concepts *Skepticism and Hype, Distance Factor, Lack of Time Commitment, Dealing with Large Customers, Fixed-bid Contracts*, and *Ineffective Customer Representative* gave rise to the category *Lack of Customer Involvement*. These concepts help describe the category *Lack of Customer Involvement* and are referred to as its *properties* [39]. Fig. 1.a shows the levels of data abstraction using GT and Fig. 1.b illustrates how the category *Lack of Customer Involvement* emerged from underlying concepts.

Another set of concepts uncovered from the analysis include *Changing Customers' Mindsets, Providing Options, Buffering, Changing Priority, Risk Assessment Up-Front, Story Owners, Customer Proxy, Just Demos, E-collaboration* and *Extreme Undercover*. These concepts led to the emergence of the category *Agile Undercover*. We analyzed the observations and compared them to the concepts derived from the interviews. We found our observations did not contradict but rather supported the data provided in interviews, thereby strengthening the interview data.

The end of open coding is marked by the emergence of a *Core* category [39]. The core category is the "main theme" or "main concern or problem" for the participants [30]. The core is the category that "*accounts for a large portion of the variation in a pattern of behaviour*" [30]. There are several criteria for choosing a category as the core. Some of these criteria are: it must be central and related to several other categories and their properties; it must reoccur frequently in the data; it takes the longest to saturate; it relates meaningfully and easily with other categories. We found the category that passed all the criteria for core was "becoming a self-organizing team".

Usually, it is not possible to summarize the entire GT research findings in a single paper and so we found it useful to focus on different aspects of the results in dedicated papers. In this paper, we present the impact of lack of customer collaboration, an important environmental factor that influences the self-organizing Agile team.

3.2.2. Selective coding

Once the core category is established, the researcher ceases open coding and moves into Selective Coding, a process which involves selectively coding for the core variable by delimiting the coding to "*only those variables that relate to the core variable in sufficiently significant ways as to produce a parsimonious theory*" [30,40]. Since lack of customer collaboration was an important factor effecting the core category — *becoming a self-organizing Agile team* — we continued to selectively code for it.

3.2.3. Memoing and sorting

The use of *memoing* — theorizing write-up of ideas about codes and their relationships — was vital in recording the relationships between codes [30]. Memos are free-flowing ideas about the codes and their relationships and they are not mixed with data. We wrote memos as and when we had ideas about the emerging codes and their relationships.

Once the data collection is nearly finished and coding is almost saturated, we proceeded to conceptually sort the theoretical memos. Sorting of the memos forms a theoretical outline, showing relationships between concepts.

3.2.4. Theoretical coding

The final step of GT is generating a theory, also known as *theoretical coding*. Theoretical coding involves conceptualizing how the categories (and their properties) relate to each other as a hypotheses to be integrated into a theory [30]. Although theoretical codes are not strictly necessary, but "*a GT is best when they are used.*" [31]. Following Glaser's recommendation, we employed theoretical coding at the later stages of analysis [39], rather than being enforced as a coding paradigm from the beginning as advocated by Strauss [41].

Glaser lists several common structures of theories known as *theoretical coding families* [30,31]. By comparing our data with the theoretical coding families, it emerged that the coding family

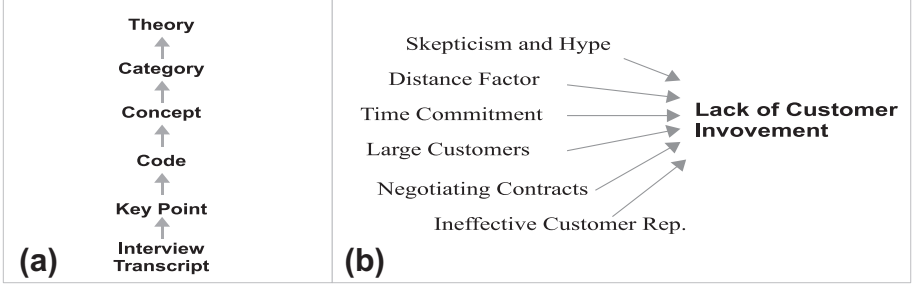

Fig. 1. (a) Levels of data abstraction in GT. (b) Emergence of category *Lack of Customer Involvement* from concepts.

best 'fit' for our data was the *Six C's* coding family [30,31,42] which describes a category in terms of its Contexts, Conditions, Causes, Consequences, Contingencies, and Covariances.

In the following section, we describe our results — the impact of inadequate customer involvement on self-organizing Agile teams — in terms of the Six C's theoretical model for the category *Lack of Customer Involvement*. Using the Six C's model, we describe (1) Contexts: the ambiance, that is, the context of the Agile development teams in NZ and India, (2) Conditions: factors that are prerequisites for the category, *Lack of Customer Involvement*, to manifest, (3) Causes: reasons that cause lack of customer involvement, (4) Consequences: outcomes or effects of lack of customer involvement on self-organizing Agile teams, (5) Contingencies: moderating factors between causes and consequences, that is, *Agile Undercover* strategies, and (6) Covariances: correlations between different categories, that is, how *Agile Undercover* strategies change when factors that cause *Lack of Customer Involvement* change.

4. Results

In the following sections we present our theory. We have adapted Glaser's Six C's model diagram [30] to illustrate our theory of lack of customer involvement (Fig. 2). The category *Lack of Customer Involvement* is at the center of the diagram. Each of the Six C's are represented in the other rectangles in relation to the central category, with corresponding subsection numbers (in circles) where we describe them.

In the following sections, we have selected quotations drawn from our interviews that shed particular light on the concepts. Due to space reasons we cannot describe *all* the underlying key points, codes, and concepts from our interviews and observation that further ground the discussion.

4.1. Context

We interviewed 30 Agile practitioners from 16 different software development organizations over 3 years, half of whom where from New Zealand and half from India. Fig. 3 shows the participants and project details. In order to respect their confidentiality, we refer to the participants by numbers P1–P30. All the teams were using Agile methods, primarily combinations of Scrum and eXtreme Programming (XP) — two of the most popular Agile methods today [14,19,13]. The teams practiced Agile practices such as iterative development, daily stand-ups, release and iteration planning, test driven-development (TDD), continuous integration and others. Participants' organizations offered products and services such as web-based applications, front and back-office applications, and local and off-shored software development services.

The level of Agile experience varied across the different teams. While some teams had under a year of experience, others had been practicing Agile for over 5 years. The Indian teams were mostly catering to off-shored customers in Europe and USA and most of the NZ teams were catering to in-house customers, some of whom were located in separate cities. We include more details of the context in sections below as necessary.

4.2. Condition

Agile projects expand the role of the customer in software development by involving them in writing user stories, discussing product features, prioritizing the feature lists, and providing rapid feedback to the development team on a regular basis [11,1,9,4]. The level of customer involvement that Agile demands is higher than their involvement in traditional projects:

> "*Commitment for that time from business... that's something that isn't normally there in a [traditional] software development project because [customers] throw [the project] over the wall and don't have to worry about it for 6 months!*" — P5, Agile Coach, NZ

Most participants did not receive the level of customer involvement that Agile methods demand (P1–P12, P14–P19, P21–P23, P25, P26, P28–P30).

> "*Sometimes [customers] only want to come back and see in 6 months what happened [in development].*" — P16, Developer, India

> "*To get client involved in the process I think is the most difficult part of Agile.*" — P4, Developer, NZ

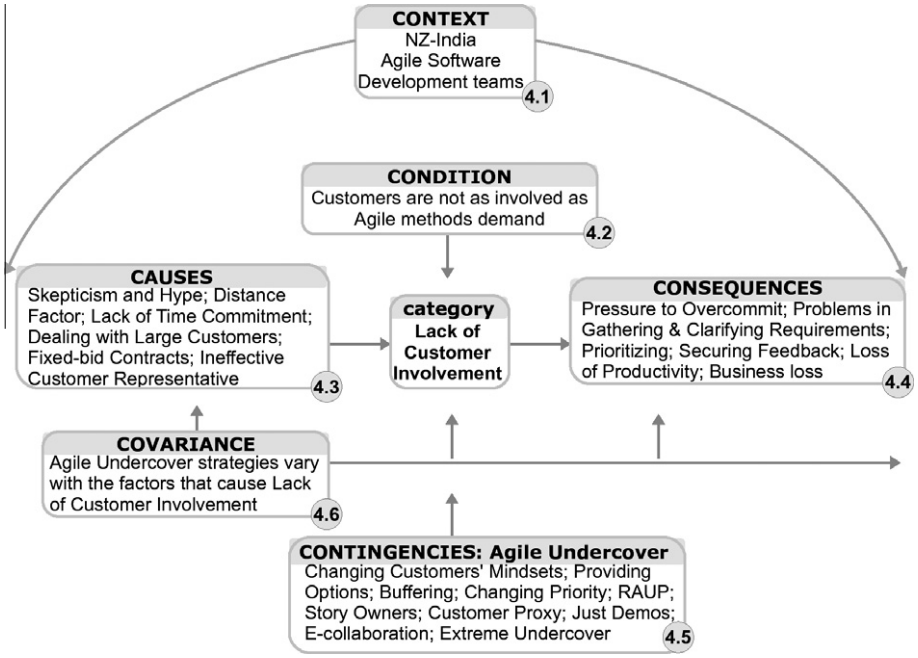

Fig. 2. The theory of *Lack of Customer Involvement* depicted using the Six C's model (Context, Condition, Causes, Consequences, Contingencies, and Covariance).

Partipants	Agile Position	Agile Method	Org. Size	Country	Domain	Team Size	Project Duration	Iteration
P1	SM	Scrum & XP	S	NZ	E-commerce	4	2	4
P2-P4	Dev × 3	Scrum & XP	S	NZ	Environment	4 to 6	12	1
P5	AC	Scrum & XP	L	NZ	Social Services	4 to 10	3 to 12	2
P6	Cust Rep	Scrum	XS	NZ	Entertainment	6 to 8	9	4
P7-P12	AC, BA, Tester, Dev × 2, Cust Rep	Scrum	M	NZ	Health	7	9	2
P13	AC	Scrum & XP	XL	NZ	Telecom & Transportation	6 to 15	12	4
P14	AC	Scrum & XP	S	NZ	Government Education	4 to 9	4	2
P15	Dev	Scrum & XP	XS	NZ	Software Development	7	6	2
P16-P20	Dev × 3, SM × 2	Scrum & XP	S	India	Software Development & Consultancy	5	6	2
P21	AC	Scrum & XP	L	India	Telecom	8 to 15	3	4
P22-P25	AC × 4	Scrum & XP	M	India	Software Development	7 to 8	3 to 6	2
P26	AC	Scrum & XP	S	India	IT & Agile Training	7 to 8	48	3
P27	Designer	Scrum & XP	S	India	Web-based services	5	1	2
P28	AC	Scrum & XP	M	India	Financial Services	8 to 11	36	2
P29	AT	Scrum	XS	India	Agile Training	7	8	3
P30	BA	Scrum & XP	M	India	Software Development	15	12	1

Fig. 3. Participant and project contexts. (Agile position: Agile Coach (AC), Developer (Dev), Customer Rep (Cust Rep), Business Analyst (BA), Senior Management (SM), Agile Trainer (AT); organizational size: XS < 50, S < 500, M < 5000, L < 50,000, XL < 100,000 employees; project duration is in months & iterations are in weeks.)

Lack of customer involved was seen as "*the most difficult part of Agile*" and "*the biggest problem*" because "*Agile [requires] fairly strong customer involvement*" (P4, P17, P30).

4.3. Causes

The participants identified several causes that had led to varying levels of customer involvement on their projects. These include: Skepticism and Hype, the Distance Factor, Lack of Time Commitment, Dealing with Large Customers, Fixed-bid Contracts, and Ineffective Customer Representative. In the following subsections, we describe each of these causes in detail, focusing on how they led to inadequate customer involvement.

4.3.1. Skepticism and hype

Agile teams often face customers that are used to traditional ways of software development. These customers would pose resistance to change and involvement. Customers harboured skepticism or misconceptions about Agile methods leading to resistance towards collaboration:

> "*typically bigger companies... would have read about it or have the wrong idea of Agile... our biggest constraint is our [customers]*" — P23, Agile Coach, India

Some Agile concepts are not readily understood and accepted by customers as originally intended. One such concept is 'fail fast' [14] which suggests that use of Agile methods will help identify projects bound to fail, quicker. The intended benefits of this — that minimum time and effort may be wasted on such a project — are not always apparent to the customer, who then becomes prejudiced about Agile:

> "*Agile terms... say option to fail early — believe me customers don't want to hear it. Customers don't want to admit that there could be some problem with the nice idea they put on paper. Forget about fail early, we dont want to fail at all!*" — P19, Senior Management, India

On the other extreme, with the increasing popularity of Agile methods, some customers treat Agile as a buzzword and are eager to reap the benefits of Agile projects without fully understanding their own responsibilities of collaboration and involvement:

> "*I mostly work [with] Indian companies with client in US... The client reads [Scrum books] and what they see is client can make changes all the time and they think wow that sounds great!... They don't understand the counter-balancing discipline [customer involvement]... Customer involvement is poor.*" — P29, Agile Trainer, India

As discussed above, practitioners mentioned both skepticism and hype as a major cause for lack of customer involvement (P5, P11, P14, P16, P17, P19, P22, P26, P29).

4.3.2. The distance factor

Distance between the Agile teams and their customers was found to be an important factor leading to lack of customer involvement (P8–P12, P17, P29).

For the outsourced teams located in India, the distance factor was multiplied manifolds as their customers were located mostly in Europe and USA:

> "*Customer involvement is poor, very adversarial relationship. Basically the customers big fear is being cheated — because they are far away, they don't know the team — every mistake seems like an indication of incompetence or vendors trying to deceive [them].*" — P29, Agile Trainer, India

The majority of the Indian participants were catering to offshored customers which made customer collaboration challenging for them due to geographic and timezone differences. As we analyzed the data from NZ participants, it emerged that outsourcing was not a specific reason behind lack of customer involvement, but rather it was the physical distance between the development team and their customers. This distinction was apparent for one of the NZ teams that had two customers, one local and the other

distant. There was a huge difference between the relationships that the team shared with the local customers versus the distant ones:

"*Relationship with [distant customer] is very different from one in [same city]. We can call at short notice... and can we borrow somebody for half a day, they are willing to do it.*" — P9, Tester, NZ

"*To be completely honest, I think the involvement came down to the fact that [local city] was [local city]. They're right across the street. They can simply come over, sit with us, we can discuss somethingface-to-face and we can really show it to [them].*" — P8, Business Analyst, NZ

"*I'm pretty sure [distance] was a problem. Because there are a lot of things that they misunderstand, or that we misunderstand.*" — P11, Developer, NZ

The team found their local customer representatives to be lot more accessible and easily available for collaboration compared to their distant counterparts. Distance between the team and their customers promoted misunderstandings and caused lack of customer involvement due to problems of communicating and coordinating over distances.

4.3.3. Lack of time commitment

Agile teams complain of not receiving enough collaboration time from their customer representatives.

"*[customers] can talk about high level 'I want to have Taj Mahal' but of granite or marble? They don't even have time to talk about that!*" — P25, Agile Coach, India

Agile teams realize that being a customer representative is a demanding responsibility often requiring extensive amount of time to be dedicated towards collaboration.

"*Big, big issues are getting enough collaboration time with the [customers]... there's no way less than a full time person would be able to keep up with getting all the requirements.*" — P5, Agile Coach, NZ

While Agile teams feel the need for more collaboration time, they also appreciate that the customer representative as their normal job responsibilities to attend to alongside the Agile project demands and their operational jobs times may take precedence:

"*More would have been better. This is the problem, when you've got someone who's got a full-time job managing a team and is quite busy, trying to fit us in.*" — P7, Agile Coach, New Zealand

Development teams feel the ability of the customer rep to devote time for collaboration is dependent on the product owner.

"*I don't think it was willingness I think it was just time [limitations]... I've never worked on [a project] where customer representative was given enough time to really be able to do the amount that they should.*" — P2, Developer, NZ

Eliciting adequate time from customers was challenging for the Agile teams and a cause for lack of customer involvement (P2, P4, P5, P7, P15, P19, P25, P26).

4.3.4. Dealing with large customers

Large customers and customers with larger projects showed a preference for traditional ways of working and were unwilling to collaborate as Agile customers. Large customers were often not bothered about the internal development-level practices of smaller Agile vendor companies and tried to enforce their traditional style of working on the team.

"*...in none of the [three] cases the customer was aware of Agile, they didn't really want to do Agile, and because of their size they were running hundreds of projects, they didn't want to care that this small organization was talking about, they just wanted to have things done in their own way.*" — P17, Developer, India

Similarly, practitioner P16 found it easier to elicit involvement from smaller customers than from larger ones:

"*Larger the organization, they are a bit less flexible towards trying out new things ... if they haven't [tried Agile before] then definitely its difficult to sell [to] them, more difficult than [to] smaller organization... in larger projects, even if they are ready for Agile they want to see atleast some milestones ... don't want to fail ... don't want to spend time and money and 2 years down the line 'we don't want Agile' ... Small projects — they say let's try Agile.*" — P16, Developer, India

Agile teams within smaller companies were unable to assert their own Agile identity and practices to the much larger customer organizations. We found evidence of customer size affecting customer collaboration only in India and not in New Zealand.

4.3.5. Fixed-bid contracts

The Agile Manifesto values "customer collaboration over contract negotiation" [8]. However in our Grounded Theory study we found that many of our Indian participants struggled with their customers' demand to fix time, cost and scope in a fixed-bid contract. The practitioners explained that the customers perceived the fixed-bid contract to give them predictability and control over the project schedule, cost, and deliverables. Since software development teams and their customers need legal contracts, and the market was such that customers could move to different software development companies if they wanted, this left our practitioners to handle the apparent contradiction between the customers' desire for *certainty* with their own commitment to Agile values such as *responding to change* [5].

"*sometimes limitations are imposed by customers, like... contracts ... they just want to give you scope, requirements and expect you to deliver it or they are looking for a fixed price contract... if you ask me biggest problems... one is contracts... they want three things: fixed deadline, fixed price, and fixed scope.*" — P19, Senior Management, India

Agile practitioners see fixed-bid contracts as a major limitation that the customers impose on them. Other practitioners shared their frustration over the issue of dealing with fixed time/scope/cost contracts.

"*Fixed price doesn't work well with Agile.*" — P21, Agile Coach, India

"*With Agile it's difficult to do fixed price projects. Agile talks about embracing change, can't do fixed price projects with changes coming in.*" — P17, Developer, India

4.3.6. Ineffective customer representative

While Indian Agile teams had limited face-to-face interactions with their customers, some NZ teams had a customer employee assigned to the project as a *customer rep*. An effective customer rep was described as *"someone who understands the implications of that system... where it fits into the business process"* and at the very least *"someone who knows how to use a computer!"* (P5, P9). Some NZ practitioners found their respective customer reps to be ineffective in providing timely requirements and feedback, while others found them lacking in proper understanding of Agile practices. They struggled with inexperienced customer reps or reps who "*haven't always understood how Scrum works*" (P8)

"*[The customer representative] didn't have any experience with the process at all... we very much had to stick with that one person.*

We couldn't say this person isn't helping us, they're not very good... it was a bit of impairment, certainly... [the team would] be sitting there for two weeks waiting for an answer." — P8, Business Analyst, NZ

The teams had little influence over who was appointed as their customer representative. In cases where the customer representative was ineffective, the teams were not able to openly raise the issue with the customer organizations.

"*Unfortunately the person who is [the customer rep] has an I.Q. of literally 25... doesn't really know how the current system works, doesn't know much about the business process, is petrified of the project sponsor, and is basically budget-driven. So she doesn't really care if it's not going to work in a way that the end users like.*" (undisclosed) Developer

Several NZ participants expressed their frustration over not being able to choose the customer reps (P2, P7, P8, P9). It is not enough to have a customer rep for the project, it is also important for that rep to be effective in providing requirements and feedback to the team. An ineffective customer rep was a contributing factor to the Agile teams wanting more or better customer collaboration.

4.4. Consequences

Inadequate customer involvement on these Agile projects led to adverse consequences for the self-organizing Agile teams. These include: Pressure to Over-commit, Problems in Gathering and Clarifying Requirements, Problems in Prioritizing Requirements, Problems in Securing Feedback, Loss of Productivity, and in extreme cases, Business Loss. We describe each of these adverse consequences of lack of customer involvement in the following subsections, focusing on how they effected the self-organizing Agile teams.

4.4.1. Pressure to over-commit

A fixed-bid contract puts the development team under pressure to deliver to the fixed constraints in the contract. The negative consequences of a fixed-bid contract in an Agile project is captured in the following comment by an Agile trainer and coach who worked mostly with Indian organizations.

"*The whole premise of the fixed-bid contract is that requirements will be fixed. The nature of software development is that requirements are inherently unstable and so when you are entering into contract negotiation, you are dealing with the recognition that the requirements will be unstable... Biggest source of dysfunction is not actually from customer — the greater source of dysfunction comes from within the organization where the contract — fixed-bid contract — is negotiated by sales team, it is negotiated for the smallest amount of money possible. And so the team from day one is under pressure to over-commit and under-deliver and that I see again and again.*" — P29, Agile Trainer, India

4.4.2. Problems in gathering and clarifying requirements

Customer representatives are meant to provide requirements in form of user stories every iteration [13]. They are also responsible for clarifying and prioritizing these stories for development. In real-life Agile projects, however, development teams faced challenges in retrieving requirements from customers:

"*To get requirements from the [customers]... was one of the worst things in this project, honestly! We'd be sitting there for two weeks waiting for an answer.*" — P8, Business Analyst, NZ

"*The biggest frustration I had on this project was that... we don't have the [customer representatives] that we can gather requirements from.*" — P10, Developer, NZ

Similarly, teams had issues trying to get customer representatives to clarify requirements:

"*Things [awaiting clarification] would queue up for them and then they'd just answer the whole queue at once... then as soon as they got busy again it would start to get a bit harder.*" — P2, Developer, NZ

As a result of insufficient and ineffective customer involvement, the development teams were unable to gather requirements and to get customer reps to clarify them for development to commence.

4.4.3. Problems in prioritizing requirements

Agile requires the customer representative to be able to prioritize the order in which the team should work on the user stories, driven by business value. Understanding and using the concept of prioritization doesn't always come naturally to customers new to Agile projects:

"*We're meant to have one list of product backlog and it's supposed to be prioritized but when the client says 'Oh that's all priority' we have to go back and say 'which?! what do you mean?!... you can't have all priority!'*" — P11, Developer, NZ

"*We'd just get a whole lot of requests sent at us, by phone and email and all different ways and it took a long, long, long time for them to understand that we needed them prioritised so we knew what was the most important to be doing.*" — P7, Agile Coach, NZ

"*[customers have to be involved... the customer needs to tell his priorities that this is the first thing we want.*" — P16, Developer, India

Providing and clarifying requirements is not enough, the customers are also required to prioritize them in order of business value. Teams faced difficulties in getting customer representatives to prioritize the requirements and as such they were confused about what features to develop and deliver first.

4.4.4. Problems in securing feedback

Customer feedback is of vital importance in ensuring the desired product is being developed and delivered incrementally. As a senior developer pointed out "*the whole point of the two week iterations was so that the end users could know if we were on the right track*" (P15) and required customer reps providing feedback on developed features.

"*If [the customer reps] didn't respond you just didn't care about their opinion... and at the end of the project... the business units that didn't give much feedback, when it went to a user, started complaining. And it's like well if we didn't get any critique it's not really our fault!*" — P2, Developer, NZ

In absence of customer feedback, teams were unable to assess how well the features met the requirements.

4.4.5. Loss of productivity

Inability to gather requirements in time for the iterations could result in unnecessary delays and loss of productivity:

"*We are extracting our requirements just in time from the business — the detailed requirements. It would be impossible if there was no full time person inside the project it would get stalled.*" — P5, Agile Coach, NZ

"*The team has the capacity... [but] with Agile if you don't have the requirement you can't do anything... because you are supposed to be in-line with business.*" — P10, Developer, NZ

Without clear requirements, the teams were forced to make assumptions about the customer's needs and priorities:

"*In the absence of business requirements from customers, the teams make assumptions and get mis-aligned from the desired business drivers. The result is a product or feature that is not aligned to the perceived business requirements.*" — P5, Agile Coach, NZ

These inaccurate assumptions led to the team building features that were not as per the customer's intended requirements. The teams would then have to perform re-work which incurred additional costs for the customers.

"*So from my perspective as a developer, yes, the more the client is involved, the better for us... But I've seen projects in the past where we had to redo all the components and it was very expensive basically to the client because we were being paid [for rework]*" — P4, Developer, NZ

Rework is both costly to customers and taxing for developers if it has to be done much later than when they developed it. Due to delay in customer feedback time, the need for re-work did not surface until much later by which time it was difficult for the developers to return to a particular story and re-work it.

"*Yes [we had to rework] but it's not the re-work, it's re-worked easily as long as it's near the time you did it. So having to go back and augment what you did three weeks ago was [hard].*" — P2, Developer, NZ

4.4.6. Business loss

The most extreme consequence of lack of customer involvement was business loss that the vendor organization suffered. Some customers were too skeptical and did not want to be involved as an Agile customer. The Agile vendor organizations, in such cases, decided to suffer business loss over working on an Agile project with no customer involvement.

"*No match between what Agile says and the way they [customers] wanted. Yes, we lost business.*" — P17, Developer, India

"*We've lost business multiple times... We try to find out early on if [Agile is] gonna be a problem and if it is, we say 'Okay, lets go our separate ways.'*" — P1, Senior Management, NZ

4.5. Contingencies: Agile Undercover

We have discussed the causes and consequences of lack of customer involvement. We now describe several interesting strategies that were used by participants to overcome the lack of customer involvement in varying degrees. These strategies are collectively named *'Agile Undercover'*, a category that emerged as explained in Section 2. These strategies include: Changing Customers' Mindsets, Providing Options, Buffering, Changing Priority, Risk Assessment Up-Front, Story Owners, Customer Proxy, Just Demos, E-collaboration, and Extreme Undercover. In the following subsections, we describe each of these strategies, focusing on how their use allowed Agile teams to to continue their Agile practices despite insufficient or ineffective customer involvement.

4.5.1. Changing customers' mindsets

"*All they [customers] have done is fixed price for last 20 years... very difficult to say it will not be fixed price.*" — P17, Developer, India

Customers are used to fixed price/scope/time contracts. Our participants disclosed that it was difficult for their customers to change their ways of working to suit Agile projects. In a bid to resolve this issue of rigid mindsets, Agile practitioners often discuss the disadvantages of traditional contracts and the advantages of Agile development methods with customers. The same practitioner P5 shared the following property of Agile practices as an advantage to customers:

"*... focus is on delivering business value as soon as possible — as a result of that you take items which are most required from point of view of business, not the ones that are most interesting in terms of technical implementation.*" — P17, Developer, India

Participant P8 noted that they often discuss with the customers how many features are seldom used. They also highlight how Agile allows the customer to avoid such situations by using "*prioritization of features*", giving them more control of the product. Agile practitioners make an effort to change the mindset of the customers and encourage them to look beyond the constraints of contracts and look at the bigger picture and the increased product control that Agile offers.

4.5.2. Providing options

Our participants shared with us some of the strategies they used to deal with the customers' expectation of fixed-bid contracts [5]. Agile practitioners offer different contract options to customers in order to encourage them to try Agile. Practitioners P3 and P8 encouraged customers to buy a few iterations to begin with instead of signing a contract for a large project up front:

"*Most of the time... [we] sell a certain number of iterations.*" — P19, Senior Management, India

By allowing the customers to use Agile on a trial basis, Agile practitioners are able to build confidence among customers and provide them with risk coverage. Once the customers have tried a few iterations, then they are offered the option to buy more iterations or features as needed:

"*One thing we [development firm] used to do and worked very well - we used to tell the customers you don't have any risks... in case of Agile we enter into a contract with the client — OK we'll show you working software every fifteen days, you'll have the option of ending the project within one sprint's notice. Maximum they can lose is one sprint. Advantage we show to client they don't have to make up their entire mind... [they] can include changes in sprints -they see it as a huge benefit to them.*" — P17, Developer, India

"*Try for a month — then buy more sprints.*" — P28, Agile Coach, India

Some Agile practitioners allow the customers to swap features. The project is delivered at the same time and price as initially specified in the contract, but the customer can remove product features that they no longer require and replace them with new ones that are of more value to them.

"*...customer after seeing demo after 4th iteration realizes the features built, say the 13th feature, is not required and he needs something else... he can swap the two.*" — P17, Developer, India

The practitioners also provide the customers with a termination clause in the contract such that customers have the option to quit on a few iterations' notice.

"*...[customers are] open to suggestions to retreat after few sprints.*" — P22, Agile Coach, India

"*[Developers] start working on functionality from day one and you can add a sprint — not enter into contract for entire project — end in one sprint's notice and they [customers] can introduce change*" — P17, Developer, India

By providing the customers with the option to quit the project in the worst case scenario, some of their financial risks were covered. So if the customers were unhappy with the results, they could always quit the project.

4.5.3. Buffering

Another practice used to adapt to the context of fixed-bid contracts was *Buffering*, which involved adding a buffer to the estimated time taken to complete a project or feature. Based on the rate of development per iteration — team velocity — as a guideline, estimates can be made about how long a particular set of requirements in a given domain will take to be developed. Then some amount of extra time was added to the estimated time as buffer. The contract is then drawn on this estimated time (including buffer) for a fixed price and scope.

> "*Agile will not ask you in how much time will you complete the project... but [the customer will]. Sometimes you've got to map internal Agile practices to customer practices... Actually it comes from a lot of experience on Agile. When you know that okay this is generally the velocity of the team that the team is able to do within the given domain, the given complexity and then you make some rough estimates, including some buffer. [Customer says] 'okay I want these features, tell me the time'. so then we'll make prediction based on Agile data that this is the team size, this is the velocity, we assume the team won't change then the Agile burndown chart will say let's say 2 weeks so we'll say okay another 2 days of buffer, so 2 weeks ands 2 days, something like that.*" — P16, Developer, India

The small amount of buffer time was important to allow the customer to the possibility of introducing changes in requirements along the way while giving the development team time to respond to those changes. Buffering was a practical strategy of working with a fixed-bid contract while using Agile methods.

4.5.4. Changing priority

In an effort to maintain the iterative and incremental nature of their Agile projects, teams were forced to change priority of user stories that were awaiting customer requirements, clarification, or prioritization. Such stories were usually demoted in priority and pushed further down into the product backlog until the required customer response was secured and development on those stories could re-commence. Agile teams confessed to that they changed the priority of the story in absence of enough, clear, and prompt requirements (P2, P8, P14, P26).

> "*[If] we know exactly what business want or we know eighty percent of what they want, we include that story in the sprint; otherwise if we have something that's a little bit unsure, we don't include that in the sprint.*" — P10, Developer, NZ

A similar strategy, called *definition of ready* [43] was adopted by an Indian team:

> "*We have recently started using... the definition of ready... product owner will not take something that is not 'done' and similarly developers are not going to take something that's not 'ready'.*" — P18, Developer, India

A user story was considered *ready* when the customers had provided the business goals and expected outcome associated with the story and implementation details necessary to estimate the story had been discussed. A story that was not *ready* was not able to achieve priority in the product backlog.

4.5.5. Risk assessment up front

One of the participants mentioned using an Agile risk assessment questionnaire as a basis to gauge the level of customer involvement on the project up front. The questionnaire included questions such as 'what is the commitment of the customer rep in terms of time?' and had multi-choice answers:

> "'*They're either assigned to this project or available as a first priority' [is] the best situation and the worst situation [is] 'Just as time allows'*" — P14, Senior Agile Coach, NZ

Performing risk assessment upfront before the start of the project allows the team to discover if the indicated level of customer involvement is a potential risk to the Agile project. Usually the reason behind limited involvement is lack of funding or unavailability of the customer rep. The approach taken to overcome this problem was to negotiate with the customer for freeing up the customer rep's time by providing funding from the project. The aim was to "*allow [the rep] to become a project team member, in a more permanent way for the duration of the project*" (P14) which is the ideal level of customer collaboration on Agile projects [14,7,13]

4.5.6. Story owners

The practice of assigning story owners was an adaptation to the Scrum practice of allocating a product owner [13]. Story owners were responsible for particular stories (less than a week long), instead of *all* the stories in the product backlog: "*every story had to have an owner to get into prioritisation.*" (P14) Assigning story owners served a three-fold purpose. Firstly, having multiple story owners instead of a single customer rep for entire project meant no one person from the customer's organization was expected to be continuously available.

> "*We didn't need that story owner for the duration of the project, we normally only need them for part of an iteration.*" — P14, Senior Agile Coach, NZ

Secondly, it allowed the team to plan out stories for development in synchronization with the corresponding story-owner's availability. Thirdly, it encouraged a sense of ownership among customer reps as they were encouraged to present their own stories to peers at end of iteration reviews.

> "*We get the [story owners] to demonstrate those stories to their peers at the end of the iteration review, this concept is something we've evolved over the project.*" — P14, Senior Agile Coach, NZ

After one such presentation a particularly skeptical customer rep was "*quite chuffed [pleased], and at the [next] iteration planning meeting, that person was all go! Instead of sitting back with their arms folded, they had their elbows on the table, leaning forward, and were driving the story detailing conversations we were having.*" (P14)

4.5.7. Customer proxy

Some Agile teams used Customer proxy — a member of the development team co-ordinating with the customers — to collaborate with customers and provide requirements and feedback. Proxies also functioned to clarify requirements from customers on behalf of the team. The use of proxy was visible in Indian teams where the customers were physically distant.

> "*Some customers say 'okay we know you do something called Agile... we can help you prioritize the backlog but we really don't know the terms like user stories and all', then we have someone called proxy customer, who really understands the customers who is there near to [them], then he is able to place their view point... some customers are open enough to provide such a person otherwise we provide... [team is happy with proxy] as long as he is able to... get the answers... it is [hard being proxy] specially when customer is on other side, there are time differences, the proxy customer has lot of things to do.*" — P16, Developer, India

"*Using Client proxy, so we assign a customer representative who interacts with the team much often but then passes on the feedback from the customer to the team and vice versa.*" — P28, Agile Coach, NZ

"*If customer had problems being product owner — we take it up on their behalf.*" — P22, Agile Coach, NZ

The use of proxy to co-ordinate between the customers and the team was also observed in New Zealand, where a Business Analyst and couple of developers on different teams were serving as the proxies.

The concept of customer proxy also proved useful for warning the team of upcoming changes from the customer's end:

"*[The proxy] still needs to get all the requirements to us, so whenever the business owner wants to make a change we know that thats going to change and... having the BA as a proxy there, a kind of buffer, you dont land that hard. And we can plan a little bit ahead.*" — P10, Developer, NZ

4.5.8. Just demos

Demonstrations are used by Agile teams as a powerful mechanism to secure the much needed and elusive customer feedback. The team presents working software to the customer representatives at these regular demonstration meetings and receive feedback from them regarding the features delivered in that iteration. This feedback is then incorporated into the development cycles.

"*Often there's someone from each of the [customers] have a look to see what were doing and how it will affect them.*" — P7, Agile Coach, NZ

Using demos provides the opportunity to clear assumptions made by development teams as a consequence of the customer representative not providing enough or clear requirements:

"*you are communicating more generally with the client by virtue of the fact that if nothing else you are releasing software more frequently in iterations to the client... Developers have their interpretations of what is that they are supposed to be doing. What we try to do to mitigate that is frequent working software that we get in front of the client and we say this is what we think you want and they say that's not even close! And we say okay cool, at least we know that now rather than at the end of the project.*" — P1, Senior Management, NZ

Demonstrations were often the only regular involvement that these Agile teams received from the customer representatives. They used this opportunity to received feedback and clarifications. The customers appreciated the demos despite their reservations about Agile in general because it provided them with increments of working software:

"*They liked what we were doing because they were not used to some additional features very fifteen days. We gave demo after fifteen days. We were getting four-five people from client organization in the demo. They were pretty impressed with that concept... happy with the results.*" — P17, Developer, India

As the local and involved customer rep of a NZ team disclosed:

"*Just the sprint demos... and [we see] three pieces of functionality and it's all done in fifteen minutes, we take the full hour to discuss the other things... the demos were fun. I dont know if thats their intent, but they were!*" — P12, Customer Rep, NZ

The demo also proved to be a useful way to get collaboration from distant customers:

"*[distant customers] can't be here every day or every week so we only got to do emailing and phonecalls during the demo.*" — P11, Developer, NZ

This strategy was found to be useful in securing customer feedback from distant and skeptical customers. Almost all customers were interested enough to attend demonstrations as it gave them an opportunity to see new functionalities of their software.

4.5.9. E-collaboration

Electronic collaboration (e-collaboration) was a popular means of communicating with customers using phone, email, chat, and voice/video conferencing. For Indian teams with off-shored customers, e-collaboration was a practical work-around:

"*Video conferencing becomes very important. Its all about collaboration [when] time difference is a problem... with Europe [there is a] 4 hours overlap.*" — P17, Developer, India

New Zealand teams were also seen using phone conferencing with shared documents and emails:

"*[Using] webX... its an online forum and as a host we get to call up documents and share them and they can come in and view.*" — P9, Tester, NZ

"*Web-conferencing... chats... [enable] stand-up meetings over the web. You can do demos that way.* — P13, Agile Coach, NZ

"*Skype or video-conferencing... doesn't cost that much — to use Skype its literally zero.*" — P10, Developer, NZ

With increasing number of software projects being off-shored globally, face-to-face collaboration has become a practical challenge. E-collaboration is a popular alternative used by software teams to overcome this issue. E-collaboration was particularly important for our Agile teams because (a) Agile requires regular customer involvement (b) several teams had physically distant customers making face-to-face collaboration difficult and (c) e-collaboration provided cheaper alternative.

4.5.10. Extreme Undercover

In an effort to avoid extreme consequences of lack of customer involvement such as business loss, Agile teams chose to follow Agile practices internally at the team level while keeping the customer unaware:

"*In none of the [three] cases the customer was aware of Agile, they didn't really want to do Agile... but what we had done was... taken charge of the projects [and] we had made it Agile — internally following Agile.*" — P17, Developer, India

Other practitioners confessed that they "*don't mention the 'A' word*" to customers who were explicitly opposed to Agile despite all the team's efforts to convince them (P28). Over the span of the research study, however, we have observed a decrease in the use of this particular practice due to increasing popularity of Agile methods among customers.

4.6. Covariance

Covariance occurs when one category changes with the changes in another category [30,42]. We found that *Agile Undercover* strategies vary with the factors that cause *Lack of Customer Involvement*. For example, *Customer Proxy*, *Just Demos* and/or *Changing Mindset* were found in practice where participants faced *Ineffective Customer Reps*. Similarly, *Customer Proxy*, *Just Demos*, E-Collaboration were found in practice where participants faced the *Distance Factor*. Fig. 4 presents the covariance relationships between *Agile Undercover* strategies being used and the factors that cause *Lack of Customer Involvement*.

Agile Undercover Strategies	Causes of Lack of Customer Involvement: Skepticism & Hype	Distance Factor	Time Commitment	Large Customers	Fixed-bid Contracts	Ineffective Customer Rep
Changing Customers' Mindsets	√			√	√	
Providing Options					√	
Buffering					√	
Changing Priority			√			
RAUP			√			
Story Owners	√		√			
Customer Proxy		√				√
Just Demos	√	√	√			√
E-Collab		√				
Extreme Undercover	√			√		

Fig. 4. Covariance between *Agile Undercover* and *Lack of Customer Involvement* (RAUP is Risk Assessment Up-Front).

5. Discussion

In the following sections, we describe the related work, implications of our results to theory and practice, and the limitations of the study.

5.1. Related work

Following classic GT, we discuss implications of *Agile Undercover* in light of existing literature after presenting the our research results [32].

Several researchers have identified varying levels of customer involvement in their own case studies, both in terms of the quality and quantity of that involvement. Conboy et al. analyzed two completed projects through the use of focus groups [44], and noted that the two teams differed dramatically in their assessment of the value of the customer's input in their project. One team consistently rated the on-site customer role as an excellent addition to their set of practices, while the second team consistently rated this role very poor — essentially counter-productive — influence on the project's successful completion. It is worth noting here that the team that rated the on-site customer role badly did so as the customer was expensive, did not actively participate in many of the key activities, and was only available for up to two hours of the typical working day due to being on a different shift. This reinforces the need for mitigating strategies where continuous and active customer collaboration cannot be achieved.

Cao et al. noted that in their multisite case study, the project managers in the observed teams acted as customer liaisons [45]. These roles were also referred to as surrogate customers, and occurred during the adoption of Agile practices, so links in with our own observations around how customer roles can be handled in situations where the team may not yet be ready for constant and active collaboration with real customers. Similarly, Mangalaraj et al. explored two projects and identified that in one project there was no dedicated customer or proxy customer, further suggesting the value of strategies to mitigate this occurrence [46]. In their project, the fact that the team had multiple external customers made it difficult for them to get sufficient buy-in, and this negatively affected the other practices that the team tried to apply.

Pikkarainen et al. [19] studied the impact of Agile practices on communication in software development and found that requirements provided by external customers were not always understandable for the developers. Korkala et al. [47] conclude that misunderstood requirements were a reason for late and unreliable software.

We found that inadequate customer involvement causes several adverse consequences for the team such as problems in gathering, clarifying, and prioritizing requirements, among other problems. Participants devised *Agile Undercover* strategies to overcome these problems. While some *Agile Undercover* strategies such as *Changing Priority, Risk Assessment Up-front, Story Owners* and *Extreme Undercover* adapted existing practices, others such as using *Customer Proxy, Just Demos* and *E-collaboration* were existing Agile practices used specifically to overcome lack of customer involvement.

Using the "definition of ready" for user stories [43] forced customers to provide detailed requirements with clear business drivers. The *definition of ready* complemented the existing Scrum definition of done [13]. The practice of assigning *Story Owners* was an adaptation of the existing product owner practice. Unlike the product owner [7,13], the story owner was only responsible for one story at a time. This was an effective way of overcoming the limited availability of customer reps. Story owners also provide an alternative to the practice of on-site customer which has been found to be effective but burdening and unsustainable for long-term use [1,7,9,47].

A *Customer Proxy* is known to be used in situations were customer involvement is not ideal [3,48,49]. Grisham et al. [1] report on the use of proxy to supplement a part-time or unavailable customer. Sometimes a proxy may work to support a Product Owner. Judy et al. describe the use of a proxy to support the Product Owner [3]. The proxy was a member of the team and an experienced Scrum Master. The use of a proxy allowed the Product Owner to fulfill their role with the minimum of time commitment and allowed the team to benefit from the continuous presence and involvement of the Product Owner proxy. Another situation where the Product Owner role may be derived from the development team is when the 'customer' is in fact the end-user. Lowery et al. report on experiences in scaling Scrum at the BBC [48]. The 'customer' in this case was the end-user of the internet services provided by BBC's online iPlayer project. As such the role of the

Product Owner was delegated to member from within the different development teams. This was akin to the use of a proxy customer that we noticed in our participants' teams. Our participants agreed that being a proxy was demanding yet useful in co-ordinating with distant customers (P10, P11, P16, P28).

Face-to-face communication is considered "*the most efficient and effective method of conveying information to and within a development*" [3,8], followed by video-conferencing, telephone, and email [47]. Our participant used *E-collaboration* extensively but noted that "*it does not take the place of having somebody sitting beside you*" (P9). Other limitations were imposed by the tool itself, such as Skype not supporting three or more people through video chatting (P10).

Although demos are a regular Agile feature, they were often the only face-to-face collaboration time our participants received from their customers and they used *Just Demos* to discuss features and receive clarifications in addition to feedback.

The customer rep is ideally an individual who has both thorough understanding of and ability to express the project requirements and the authority to take strategic decisions [11,1,4]. Boehm advocates dedicated and co-located CRACK (Collaborative, Responsible, Authorized, Committed, Knowledgeable) customers for Agile projects [50].

Martin et al. [9] found that the on-site customer role in XP projects, although perceived as rewarding by some customers, was largely seen as overburdening and inherently unsustainable. They discovered that the customer role was played by a team of people, instead of by a single person as initially assumed in literature. Martin et al. describe an informal XP customer team that consist of different roles. Of these different roles, it is the Negotiator role that is the closest to the classic customer representative role and interacts directly with the development team. The Negotiator is a customer representative that has in-depth domain knowledge, provides requirements to the development team, and is willing to carry responsibility of project success or failure. In addition, our participants suggested that customers should choose representatives that understand both Agile practices and their own responsibilities in the process of Agile software development (P5, P12, P29). Martin et al. describe certain customer practices such as Customer Boot Camp and Pair Customering. These practices — when combined with the customer roles they identified — can help reduce the burden placed on the on-site customer role and the XP team.

Extreme Undercover was used by Indian teams as a last resort, specially when facing business loss. While one NZ team faced business loss but they chose to bear it instead of practicing *Extreme Undercover*. Over the research period (3 years) we found that the use of *Extreme Undercover* diminished with the increase in popularity of Agile methods.

In an earlier paper, we described six self-organizational roles on Agile teams: *Mentor*, *Co-ordinator*, *Translator*, *Champion*, *Promoter*, and *Terminator* [33]. These roles are informal, implicit, and often temporary emerging in response to problems faced by the self-organizing Agile team. The *Mentor*, *Champion*, and *Terminator* roles focus internally on the team and their problems within their own organizational context. The *Co-ordinator*, *Translator*, and *Promoter* roles, on the other hand, are dedicated to helping the Agile team better co-ordinate and collaborate with their customers, demonstrating that (a) effective customer collaboration on Agile projects is important and (b) that securing customer collaboration can be a challenge for self-organizing Agile teams. The Promoter uses the strategy of *Changing the Customers' Mindsets* by explaining the benefits of Agile practices to their customers in an attempt to secure their involvement. The Co-ordinator is the ideal candidate to be the *Customer Proxy* when the customer representative is unable to co-ordinate with the team directly.

5.2. Implications for theory

5.2.1. Agility in context

Agile principles and practices assume a project context where the customer is both sufficiently and effectively collaborating with the development team. The results of our study show that levels of customer collaboration may vary largely with different real-life project contexts. As such, an assumption of adequate customer involvement is not only incorrect in many cases, but also leaves the development team wondering what to do when their customers do not collaborate.

Current Agile principles and practices do not cater to the context where levels of customer collaboration may not be ideal. For example, Agile principles support "*customer collaboration over negotiating contracts*" and yet we found that customers' demands for fixed-bid contracts hampered the ability of the development teams to practice Agile methods. Such problems encountered by self-organizing Agile teams in securing and maintaining adequate customer collaboration forced them to device strategies to practice Agile methods despite inadequate customer collaboration. The emergence of these *Agile Undercover* strategies suggests that we need to understand and apply Agility in light of the project contexts. We have developed a more detailed theory of adaptation in Agile development which has been described elsewhere [51].

5.2.2. Beyond Agile customers

An implication of our Grounded Theory results is towards building a formal Grounded Theory on the lack of customer involvement that is applicable outside the substantive area of Agile software development. Another possibility is conducting studies in areas of more traditional software development to explore the level of customer involvement, its consequences on the development teams, and the strategies the teams may devise to overcome any problems. Similarly, a study comparing the impact of customer collaboration in Agile methods versus that in traditional methods is possible.

5.3. Implications for practice

Our study serves to emphasize the impact of the customer collaboration on self-organizing Agile teams. As we described in our findings, inadequate customer involvement leads to adverse consequences such as pressure to over-commit, problems in gathering, clarifying, and prioritizing requirements, securing feedback, and loss of productivity and business. Customers must realize their responsibilities in ensuring the success of the self-organizing Agile projects and, in turn, that of their projects.

5.3.1. Continuum of customer involvement

Our findings reveal that there is a continuum of levels of customer involvement on real-life Agile projects (Fig. 5). Fig. 5 depicts the continuum of levels of customer involvement based on the directness (face-to-face, in person) of the collaboration. The ideal level is a most direct customer involvement via the on-site customer where the real customer representative is present face-to-face and in person for most collaboration-intensive practices as per Agile guidelines. The least desirable level is *Extreme Undercover* where there is no face-to-face or in person involvement from the customer's organization. The levels in between the two extremes are not strictly linear and may occur simultaneously, such as *Just Demos* may take place using *E-collaboration*. The continuum assumes that the amount and quality of involvement are the same for all levels.

Agile teams use some *Agile Undercover* strategies to secure the ideal level of customer involvement on their projects, such as: using *Risk Assessment Up-front* to measure the potential level of

Fig. 5. Continuum of customer involvement on Agile projects.

customer involvement on the project, *Changing Customers' Mindsets* to encourage them to get involved in the project, *Providing Options* and *Buffering* to work-around customer demands for fixed-bid contracts, *Changing Priority* of stories to force customer representatives to provide complete requirements and feedback. Despite these strategies, many teams are not able to secure on-site customer involvement.

In absence of the on-site customer, Agile teams developed different strategies to secure varying levels of customer involvement ranging from *Story Owners* where members of the customer organization share the responsibility of the customer role and are available as and when required; *Just Demos* where the level of customer involvement is limited to participating in end of iteration demonstrations; *E-collaboration* where the team interacts with the customer representative over electronic means such as videoconferencing; *Customer Proxy* from the team playing the role of the customer representative in absence of the real customer involvement; followed by the least desirable level, *Extreme Undercover* where the customer is unaware of the Agile nature of the project.

5.3.2. Extreme Undercover as a transitioning strategy

Extreme Undercover was also used by development teams as a temporary measure to transition into Agile practices. One of the software development companies who were internally moving from traditional to Agile methodologies used *Extreme Undercover* during their transition phase. Transitioning can be a significant challenge. One coping strategy was to first successfully adopt and transition into Agile at their team levels before inviting the customer to get involved in the Agile practices. During internal adoption of Agile, practitioners still have their normal project commitments to take care of alongside understanding and accepting the changes that come along with Agile adoption. Since the development organization did not want to lose business with existing customers, they chose to go undercover and adopt Agile internally as a first step. The Agile practices grew from the inside out, starting from the team level and gradually involving the customer. They chose to implement Agile practices internally and put a robust fundamental Agile framework in place before they decided to go public with their Agile status.

5.3.3. Problem-solution chart

The covariance chart (Fig. 4) has direct implications for practice: it serves as a guide for Agile practitioners to assess the problem(s) causing inadequate customer involvement in their own context, compare them to the causes listed on the chart, and try corresponding *Agile Undercover* strategies to see if they help in their contexts. In this way, the covariance chart has practical implications as a "problem-solution" chart for Agile practitioners.

5.4. Limitations

The inherent limitation of a Grounded Theory study is that the resulting theory can only be said to explain the specific contexts explored in the study. Since the codes, concepts, and category emerged directly from the data, which in turn was collected directly from real world, the results are grounded in the context of the data [52]. Those contexts were dictated by our choice of research destinations, which in turn were in some ways limited by our access to them.

As with any empirical Software Engineering, the very high number of variables that affect a real Software Engineering project may it difficult to identify the influence that any one factor has on the success or failure of the project. The impact of inadequate customer involvement on self-organizing Agile teams, however, was clearly evident.

The results presented are from participants who are all currently performing Agile practices within self-organizing Agile teams. Although we have a couple of customer representatives among the participants, none of them are end-customers, and so our strategies — and their consequences — are necessarily from the development team's perspective. This limitation is mitigated by the fact that the development team has a significant input in deciding whether Agile practices are worth following in the long-term. Therefore, analyzing strategies (and their perceived consequences) from the development team's perspective is a valid contribution, as the team must have confidence in their own practices. However, future work can complete this picture by exploring how customers react to these strategies, and whether they see them as a useful contributing factor to a project's success.

One way to evaluate an emerging theory is to present it to experts in the field to gain their feedback. When the experts in the field find the research findings useful, it can become an important source of verifying the fit, work and relevance of the theory [30]. We discussed our emerging results with several practitioners in India and New Zealand, who found our results relevant. Receiving comments such as 'rings true' and 'well applied' from the Agile practitioners made us confident of our emerging theory. Frequent discussions with the research supervisors about emerging codes, concepts, and categories, as well as frequent presentations to — and feedback from — the Agile practitioner communities in NZ and India, helped validate the emerging results.

Data derived from interviews is prone to bias [37]. We found that it takes time to build the ability and skill to be able to ask questions that can counter-check the data provided. We also ensured authenticity of the data collected through interviews by supplementing it with observations of workplaces and activities [37]. The data derived from observations did not contradict, but rather supported the interview data, thereby strengthening it.

We gathered a rounded perspective of the issues by interviewing practitioners representing other aspects of software development such as customer representative and senior management besides focusing on the development team (developer, tester, Agile coach, business analyst). In order to minimize any loss or misinterpretation, all data was personally collected and analyzed by the same researcher, namely the first author.

5.5. Future work

The continuum of customer involvement described in Fig. 5 serves as a guidelines to measure the level of customer involvement on the project. Future studies could use this guideline to build diagnostic tools to evaluate the level of customer involvement on Agile projects. The covariance chat (Fig. 4) can also serve as a basis for building tools to assess the problems causing inadequate customer involvement on Agile projects and recommending appropriate strategies to overcome the problems. The *Agile Undercover* strategies must be integrated into Agile methods in order to make it readily available to greater number of practitioners. Future studies could explore the viability and success of these strategies in different contexts such as in other countries and cultures.

6. Conclusion

We conducted a GT study, involving 30 Agile practitioners from 16 different software development organizations in New Zealand and India, over a period of 3 years. The results reveal that customers are not as involved on these Agile projects as Agile methods demand. In this paper, we have described (a) the causes of lack of customer involvement on these Agile projects, (b) its adverse consequences faced by the development teams (c) *Agile Undercover* strategies used by these Agile teams to practice Agile despite insufficient or ineffective customer involvement, (d) description of our application of Grounded Theory, including the six C's theoretical model to explain the causes, consequences, and strategies, and (e) a discussion of implications of our results in theory and practice.

Some *Agile Undercover* strategies were adapted practices while others were close to existing Agile practices. Participants found the *Agile Undercover* strategies to be largely useful and effective in their own contexts. Although we do not prescribe *Agile Undercover* strategies as replacement for real and valuable customer involvement, they may assist Agile teams facing similar lack of customer involvement.

Acknowledgments

Our thanks to all the participants. This research is generously supported by an Agile Alliance academic grant and a NZ BuildIT PhD scholarship.

References

[1] P.S. Grisham, D.E. Perry, Customer relationships and extreme programming, in: HSSE '05: Proceedings of the 2005 Workshop on Human and Social Factors of Software Engineering, ACM, New York, NY, USA, 2005, pp. 1–6.
[2] G.K. Hanssen, T.E. Faegri, Agile customer engagement: a longitudinal qualitative case study, in: ISESE '06: Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, ACM, New York, NY, USA, 2006, pp. 164–173.
[3] K.H. Judy, I. Krumins-Beens, Great scrums need great product owners: unbounded collaboration and collective product ownership, in: HICSS '08: Proceedings of the 41st Annual Hawaii International Conference on System Sciences, IEEE Computer Society, Washington, DC, USA, 2008, p. 462.
[4] S. Nerur, R. Mahapatra, G. Mangalaraj, Challenges of migrating to agile methodologies, Commun. ACM 48 (2005) 72–78.
[5] R. Hoda, J. Noble, S. Marshall, Agile undercover: when customer's don't collaborate, in: XP2010, Springer, Norway, 2010, pp. 73–87.
[6] T. Chow, D. Cao, A survey study of critical success factors in agile software projects, J. Syst. Softw. 81 (2008) 961–971.
[7] T. Dybå, T. Dingsoyr, Empirical studies of agile software development: a systematic review, Inf. Softw. Technol. 50 (2008) 833–859.
[8] J. Highsmith, M. Fowler, The agile manifesto, Softw. Dev. Mag. 9 (2001) 29–30.
[9] A. Martin, R. Biddle, J. Noble, The XP customer role: a grounded theory, in: AGILE2009, IEEE Computer Society, Chicago, 2009, pp. 33–40.
[10] S.C. Misra, V. Kumar, U. Kumar, Identifying some important success factors in adopting agile software development practices, J. Syst. Softw. 82 (2009) 1869–1890.
[11] S. Fraser, A. Martin, R. Biddle, D. Hussman, G. Miller, M. Poppendieck, L. Rising, M. Striebeck, The role of the customer in software development: the XP customer – fad or fashion?, in: OOPSLA, ACM, New York, 2004, pp 148–150.
[12] C. Larman, V.R. Basili, Iterative and incremental development: a brief history, Computer 36 (2003) 47–56.
[13] K. Schwaber, M. Beedle, Agile Software Development with SCRUM, Prentice-Hall, 2002.
[14] K. Beck, Extreme Programming Explained: Embrace Change, first ed., Addison-Wesley Professional, 1999.
[15] A. Cockburn, Crystal Clear: A Human-Powered Methodology for Small Teams, Addison-Wesley Professional, 2004.
[16] S. Palmer, M. Felsing, A Practical Guide to Feature-Driven Development, Pearson Education, 2001.
[17] J. Stapleton, Dynamic Systems Development Method, Addison Wesley, 1997.
[18] J. Highsmith, Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, Dorset House Publishing, New York, 2000.
[19] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, J. Still, The impact of agile practices on communication in software development, Empirical Softw. Eng. 13 (2008) 303–337.
[20] A. Cockburn, J. Highsmith, Agile software development: the people factor, Computer 34 (2001) 131–133.
[21] R. Martin, Agile Software Development: Principles, Patterns, and Practices, Pearson Education, NJ, 2002.
[22] K. Schwaber, Scrum Guide, Scrum Alliance Resources, <http://www.scrum.org/storage/scrumguides/ScrumGuide.pdf> (accessed September 2010).
[23] H. Sharp, H. Robinson, Collaboration and co-ordination in mature extreme programming teams, Int. J. Hum.–Comput. Stud. 66 (2008) 506–518.
[24] J. Highsmith, Agile Project Management: Creating Innovative Products, Addison-Weasley, USA, 2004.
[25] H. Takeuchi, I. Nonaka, The new product development game, Harvard Bus. Rev. 64 (1986) 137–146.
[26] S. Augustine, B. Payne, F. Sencindiver, S. Woodcock, Agile project management: steering from the edges, Commun. ACM 48 (2005) 85–89.
[27] L. Anderson, G. Alleman, K. Beck, J. Blotner, W. Cunningham, M. Poppendieck, R. Wirfs-Brock, Agile management – an oxymoron: who needs managers anyway?, in: OOPSLA '03, ACM, New York, 2003, pp 275–277.
[28] T. Chau, F. Maurer, Knowledge sharing in agile software teams, 2004.
[29] S. Fraser, R. Reinitz, J. Eckstein, J. Kerievsky, R. Mee, M. Poppendieck, Xtreme programming and agile coaching, in: OOPSLA Comp.03, ACM, New York, 2003, pp. 265–267.
[30] B. Glaser, Theoretical Sensitivity: Advances in the Methodology of Grounded Theory, Sociology Press, Mill Valley, CA, 1978.
[31] B. Glaser, The Grounded Theory Perspective III: Theoretical Coding, Sociology Press, Mill Valley, CA, 2005.
[32] B. Glaser, A.L. Strauss, The Discovery of Grounded Theory, Aldine, Chicago, 1967.
[33] R. Hoda, J. Noble, S. Marshall, Organizing self-organizing teams, in: ICSE '10: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, ACM, New York, NY, USA, 2010, pp. 285–294.
[34] A. Cockburn, People and methodologies in software development, Ph.D. thesis, University of Oslo, Norway, 2003.
[35] G. Coleman, R. OConnor, Using grounded theory to understand software process improvement: A study of irish software product companies, Inf. Softw. Technol. 49 (2007) 654–667.
[36] G. Allan, A critique of using grounded theory as a research method, in: EJBRM, vol. 2, 2003.
[37] K. Parry, Grounded theory and social process: a new direction for leadership research, Leadership Quart. 9 (1998) 85–105.
[38] S. Georgieva, G. Allan, Best practices in project management through a grounded theory lens, Electron. J. Bus. Res. Methods 6 (2008) 43–52.
[39] B. Glaser, Basics of Grounded Theory Analysis: Emergence vs Forcing, Sociology Press, Mill Valley, CA, 1992.
[40] B. Glaser, Remodeling grounded theory, Forum: Qual. Soc. Res. 5 (2004).
[41] A. Strauss, J. Corbin, Basics of Qualitative Research: Grounded Theory Procedures and Techniques, Sage Publications, Newbury Park, CA, 1990.
[42] M.M. Kan, K.W. Parry, Identifying paradox: a grounded theory of leadership in overcoming resistance to change, Leadership Quart. (2004) 467–491.
[43] S. Beaumont, The definition of ready, Xebia Blogs, 2010, <http://blog.xebia.com/2009/06/19/the-definition-of-ready>.
[44] K. Conboy, Agility from first principles: reconstructing the concept of agility in information systems development, Info. Sys. Res. 20 (2009) 329–354.
[45] L. Cao, K. Mohan, P. Xu, B. Ramesh, A framework for adapting agile development methodologies, Eur. J. Inform. Syst. 18 (2009) 332–343.
[46] G. Mangalaraj, R. Mahapatra, S. Nerur, Acceptance of software process innovations – the case of extreme programming, Eur. J. Inform. Syst. 18 (2009) 344–354.
[47] M. Korkala, P. Abrahamsson, P. Kyllonen, A case study on the impact of customer communication on defects in agile software development, in: AGILE '06: Proceedings of the Conference on AGILE 2006, IEEE Computer Society, Washington, DC, USA, 2006, pp. 76–88.
[48] M. Lowery, M. Evans, Scaling product ownership, in: AGILE '07: Proceedings of the AGILE 2007, IEEE Computer Society, Washington, DC, USA, 2007, pp. 328–333.
[49] C. Mann, F. Maurer, A case study on the impact of scrum on overtime and customer satisfaction, in: ADC '05: Proceedings of the Agile Development Conference, IEEE Computer Society, Washington, DC, USA, 2005, pp. 70–79.
[50] B. Boehm, R. Turner, Rebalancing your organization's agility and discipline, in: XP/Agile Universe '03, Springer, Berlin, 2003, pp. 1–8.
[51] R. Hoda, P. Kruchten, J. Noble, S. Marshall, Agility in context, in: SPLASH/OOPSLA2010, ACM, Reno, USA, 2010.
[52] S. Adolph, W. Hall, P. Kruchten, A methodological leg to stand on: lessons learned using grounded theory to study software development, in: CASCON '08, ACM, New York, 2008, pp. 166–178.