Organizing *Self-Organizing* Teams

Rashina Hoda
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
rashina@ecs.vuw.ac.nz

James Noble
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
kjx@ecs.vuw.ac.nz

Stuart Marshall
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
stuart@ecs.vuw.ac.nz

ABSTRACT

Agile teams are described as "self-organizing". How these teams actually organize themselves in practice, however, is not well understood. Through Grounded Theory research involving 24 Agile practitioners across 14 software organizations in New Zealand and India, we identified six informal roles that team members adopt in order to help their teams self-organize. These roles — *Mentor*, *Co-ordinator*, *Translator*, *Champion*, *Promoter*, and *Terminator* — help teams learn Agile practices, liaise with customers, maintain management support, and remove ineffective team members. Understanding these roles will help software teams become self-organizing, and should guide Agile coaches in working with Agile teams.

Categories and Subject Descriptors

K.6.1 [**Project and People Management**]: Management techniques; K.6.3 [**Software Management**]: Software Development Process

General Terms

Management, Human Factors, Theory

Keywords

Software Engineering, Self-organizing teams, Agile software development

1. INTRODUCTION

Agile software development has gained popularity in the software engineering industry [9, 36] as well as in the research community [17, 34, 36]. Although several research studies have explored the social aspects of Agile teams [13, 34, 41], there has been no substantial research on the self-organizing nature of Agile teams across multiple projects, organizations, and cultures. Our goal in this paper is to understand the self-organizing nature of Agile teams and the roles that facilitate self-organization.

ICSE '10, May 2-8 2010, Cape Town, South Africa

Self-organizing teams are at the heart of Agile software development [15, 27, 33, 39], and self-organization is one of the 12 principles behind the Agile Manifesto [27]. Self-organizing teams manage their own work, and organize around the details of the tasks [15], and can greatly influence team effectiveness [34]. The Agile Manifesto [27] claims that the best architectures, requirements, and designs emerge from self-organizing teams.

Leadership in Agile teams is distributed, providing subtle control and direction to the team [5, 11, 45] in contrast to centralized management in traditional teams [10, 19, 29]. Scrum Masters [40] and XP Coaches [8] are seen facilitators, and do not directly organize the team [18, 35]. This leaves a critical question unanswered: *How do self-organizing Agile teams organize themselves?*

In answer to this question, we found that Agile team members adopt 6 roles, specifically to support their team's self-organization: *Mentor*, *Co-ordinator*, *Translator*, *Champion*, *Promoter*, and *Terminator*. We conducted a Grounded Theory study involving 24 Agile practitioners in 14 different software organizations in New Zealand and India. Their software development projects employed Scrum, or combinations of Scrum and XP (eXtreme Progamming), two of the most popular and widely adopted Agile methods [20]. We conducted several rounds of face-to-face semi-structured interviews, observations, and data analysis over 2 years. In order to get a rounded perspective, we interviewed a wide range of project participants about their experiences working on Agile projects.

2. BACKGROUND AND RELATED WORK

Agile software development methods follow an iterative and incremental style of development where collaborative self-organizing teams dynamically adjust to changing customer requirements [1, 17, 33]. The Agile Manifesto [27] defines the four basic values of Agile methods as:

> "*Individuals and interactions over processes and tools*
> *Working software over comprehensive documentation*
> *Customer collaboration over contract negotiation*
> *Responding to change over following a plan*
> *That is, while there is value in the items on the right, we value the items on the left more.*"

Self-organizing teams are one of the twelve principles behind the manifesto [27]. There are various methods that seek to embody the Agile Manifesto including Scrum [40], XP (eXtreme Programming) [8], Crystal [14], FDD (Feature Driven Development) [37], DSDM (Dynamic Software Development Method) [43], and Adaptive Software Development [25]. Scrum and XP are considered to be the most widely adopted Agile methods in the world [20]. XP

focuses on developmental practices, while Scrum mainly covers project management [17].

2.1 Self-Organizing Teams in Agile

Agile teams are self-organizing teams [12, 15, 27, 33, 39]. Self-organizing teams are composed of "*individuals [that] manage their own workload, shift work among themselves based on need and best fit, and participate in team decision making.*" [26]. Self-organizing teams organize repeatedly to meet new challenges [15]. They manage their own work and organize around the details of their tasks. Self-organizing teams must have common focus, mutual trust, and respect [15].

Sharp and Robinson [42] note that mature XP teams are highly collaborative and self-organising in nature. Self-organization can also directly influence team effectiveness as found in another study [34] as self-management brings decision making authority to the level of operational problems, which increases the speed and accuracy of problem solving.

Takeuchi and Nonaka [45] describe self-organizing teams as exhibiting autonomy, self-transcendence, and cross-fertilization. Moe et al. [35] use one of these aspects — autonomy — to represent the concept of self-organizing teams and explore barriers to self-organization on a single project. Their study identifies division of work based on specialized skills as the most important barrier to self-organization. Moe et al. note that Scrum emphasizes self-organizing teams but does not provide clear guidelines on how they should be organized [34].

2.2 Facilitation versus Leadership

Agile methods require that the role of the project manager needs to change from being a controller and planner to becoming a facilitator and collaborator [36]. It is recommended that managers on Agile projects be ready to relinquish some control [7, 36, 45]. In a OOPSLA panel discussion [5] Poppendieck notes "*I distinguish management tasks - getting the maximum value from the dollar - from leadership tasks - helping people to excel. Leaders are required. Managers are optional.*"

Self-organizing teams are not supposed to be leaderless and uncontrolled [15, 45]. Leadership in self-organizing teams is meant to be *light-touch* and *adaptive* [7], providing feedback and subtle direction [5, 11, 45]. Leaders of Agile teams are often compared to coaches of sports teams — responsible for setting direction, aligning people, obtaining resources, and motivating the teams [5]. Agile projects have job titles such as Scrum Masters [40] and (XP) Coaches [18] instead of traditional managers.

According to the Scrum and XP guidelines, a Scrum Master is responsible for protecting the team from any disruptions to their tasks that may be caused by outside sources [34, 38, 40] such as unrealistic demands from the customers. They ensure that the team is fully functional and productive and that all Scrum processes are being followed [40]. A Scrum Master is seen as a facilitator and does not organize or manage the team [39]. Similarly, an XP Coach is meant to lead the team towards self-organization by leaving the team alone as early as possible [18].

3. RESEARCH CONTEXT AND METHOD

3.1 Participants and Projects

We interviewed 24 Agile practitioners from 14 different software organizations in 2 countries — New Zealand and India — to which we had access. We chose our participants from two cultures, not knowing how or if culture would play a part in the results.

We interviewed participants from a range of different roles within Agile projects, so as to ensure that we had a rounded perspective of how their Agile teams worked. In particular, we interviewed and observed Agile Coaches (Scrum Masters and XP Coaches), Developers, Designers, Testers, Business Analysts, Product Owners, and Senior Management. All the teams we studied have adopted Agile development methods, primarily Scrum. The teams used Agile practices such as iterative development, release and iteration planning, test-driven development, daily stand-up meetings, frequent delivery of software, and continuous integration. Table 1 shows participant and project details.

The projects' durations varied from 1 to 48 months and the team size varied from 4 to 15 people on different projects. The products and services that the participants' organizations offered included web-based applications, front and back-end functionality, and local and off-shored software development services. Half the participants were practicing in India and half in New Zealand. The organizational sizes varied from 10 to 300,000 employees. In order to respect their confidentiality, we refer to our participants by numbers P1 to P24.

3.2 Grounded Theory

Grounded Theory (GT) is the systematic generation of theory from data acquired by a rigorous qualitative research method [21, 22]. GT was developed by sociologists Glaser and Strauss [23].

We chose to use 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 suited to areas of research which have not been explored in great detail before, and the research literature on self-organizing Agile teams is scarce. Using GT, we have applied a rigorous research method to study practical applications of Agile methods and to analyze and explain the results. Finally, GT is being increasingly being used to study the social nature of Agile teams [13, 31, 47]. Using Glaser's approach, we started out with a general area of interest — Agile project management — rather than beginning with a research problem [16].

3.3 Data Collection

We conducted face-to-face, semi-structured interviews with Agile practitioners using open-ended questions. The interviews were approximately an hour long and were scheduled at the practitioners' workplaces or mutually agreed public locations. The interview questions focused on the participants' experiences of working with Agile methods and in particular around their roles on Agile projects. For example, we asked about the challenges participants faced in Agile projects and the strategies they used to overcome them. The answers varied with the individual participants, however, as we later discovered during analysis, self-organizing teams emerged as the largest and most common concern. Then, further analysis indicated that participants played characteristic roles to facilitate self-organizing teams.

The interview data was strengthened by our observation of several Agile practices on two projects in New Zealand and two in India. We attended and observed Agile practices such as daily stand-up meetings (co-located and distributed), release planning, iteration planning, and demostrations of the teams from which our interview participants were derived. We conducted data collection and analysis iteratively so that constant comparison of data helped guide future interviews and the analysis of interviews and observations fed back into the emerging results. In order to maintain consistency in the application of Grounded Theory, all data was collected and analyzed personally by the primary researcher.

Table 1: Participant and Project Details. * Organizational Size: XS < 50, S < 500, M < 5000, L < 50,000, XL > 100,000 employees

Participant	Agile Position	Agile Method	Org. Size*	Country	Domain	Team Size	Project Duration (months)	Iteration (weeks)
P1	Agile Coach	Scrum & XP	XL	NZ	Telecom & Transportation	6 to 15	12	4
P2	Agile Coach	Scrum & XP	L	NZ	Social Services	4 to 10	3 to 12	2
P3, P4, P5	Developer × 3	Scrum & XP	S	NZ	Environment	4 to 6	12	1
P6	Product Owner	Scrum	XS	NZ	Entertainment	6 to 8	9	4
P7, P8, P9, P10	Business Analyst, Tester, Developer × 2	Scrum	M	NZ	Health	7	9	2
P11	Agile Coach	Scrum & XP	S	NZ	Government Education	4 to 9	4	2
P12	Senior Management	Scrum & XP	S	NZ	E-commerce	4	2	4
P13, P14, P20	Senior Management, Developer × 2	Scrum & XP	S	India	Software Development & Consultancy	5	6	2
P15, P16, P17, P18	Agile Coach × 4	Scrum & XP (Own Version)	M	India	Software Product Development	7 to 8	3 to 6	2
P19	Agile Coach	Scrum & XP	L	India	Telecom	8 to 15	3	4
P21	Agile Coach	Scrum & XP	S	India	IT & Agile Training	7 to 8	48	3
P22	Agile Trainer	Scrum	XS	India	Agile Training	7	8	3
P23	Designer	Scrum & XP	S	India	Web-based services	5	1	2
P24	Agile Coach	Scrum & XP	M	India	Financial Services	8 to 11	36	2

3.4 Data Analysis

We used open coding to analyze the interview transcripts in detail [2, 24]. To explain the GT data analysis method, we present an example of working from interview transcripts to results for one of the roles, *Mentor*.

We began by collating key points from each interview transcript [24]. We then assigned a *code* to each key point. A *code* is a phrase that summaries the key point in 2 or 3 words.

Interview quotation: "*We had [Coach] as well at the time [the team started Agile practices] so...It made it easy...having [Coach] there as a backup...[it has] been really good to have that guidance from [Coach].*" — P8, Tester, NZ

Key Point: "*Coach providing guidance in initial stages*"

Code: Providing initial guidance (P8, NZ)

The codes arising out of each interview were constantly compared against the codes of the same interview, and those from other interviews and observations. This is GT's *constant comparison method* [22, 23]. In the example, other similar codes were "providing support (P18, IN)" and "showing people through Agile process (P2, NZ)". Using the constant comparison method we grouped these codes to produce a higher level of abstraction, called *concepts* in GT.

Concept: Providing initial guidance and support

Other concepts that emerged included: teaching Agile practices; removing misconceptions; getting the team confident in the use of Agile methods; and encouraging continued adherence to the method. Finally we repeated the constant comparison method on concepts to produce a third level of abstraction called *Categories*.

Category: Mentor

A *Mentor* is one particular individual in the Agile team that assumes the responsibility of providing guidance on the chosen Agile method. Figure 1.A illustrates how the category *Mentor* emerged from the underlying concepts and 1.B shows the levels of data abstraction using GT.

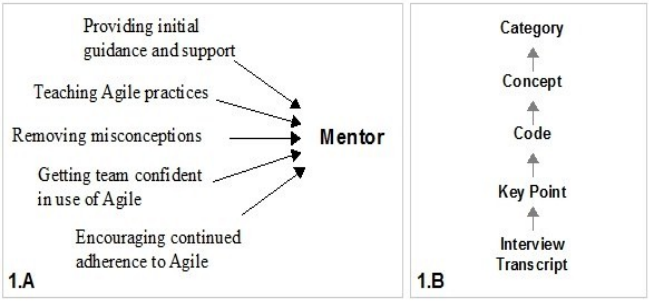

Figure 1: A: Emergence of the category *Mentor* from concepts. B: Levels of Abstraction in Grounded Theory.

Since the codes, concepts, and categories emerge directly from the data, which in turn is collected directly from the real world, the resulting theory is grounded within the context of the data [4].

The other categories emerged in a similar fashion, but we are unable to illustrate this process for all of them for space reasons. The resulting categories form the grounded theory of Agile self-organizing teams.

In the following section we present the categories of our theory, the *Mentor*, *Co-ordinator*, *Translator*, *Champion*, *Promoter*, and *Terminator* roles that facilitate self-organizing teams. We have

Table 2: Roles Facilitating Self-Organizing Agile Teams

Role	Definition	Played by	Interacts with
Mentor	Guides and supports the team initially, helps them become confident in their use of Agile methods, and encourages continued adherence to Agile practices.	Agile Coach	Team
Co-ordinator	Acts as a representative of the self-organizing Agile team to coordinate communication and change requests from customers.	Developer, Business Analyst	Team, Customers
Translator	Understands and translates between the business language used by customers and the technical terminology used by the team, in an effort to improve communication between the two.	Business Analyst	Team, Customers
Champion	Champions the Agile cause with the senior management within their organization in order to gain support for the self-organizing Agile team.	Agile Coach	Senior Management
Promoter	Promotes Agile with customers and attempts to secure their involvement and collaboration to support the efficient functioning of the self-organizing Agile team.	Agile Coach	Customers
Terminator	Identifies team members threatening the proper functioning and productivity of the self-organizing Agile team and engages senior management support in removing such members from the team.	Agile Coach	Team, Senior Management

selected quotations drawn from our interviews that shed particular light on these categories and that are spread across participants, geographically and by their organizational title. The discussion is grounded further by underlying key points, codes, and concepts from our interviews and observations, that we cannot describe in detail for space reasons.

4. RESULTS

The contribution of this paper — the key result of our study — is that we have found that team members adopt one or more of 6 informal roles to facilitate their team's self-organization, besides the traditional organizational roles. These roles are defined in Table 2. A team member may play one or more of these self-organisational roles, along with e.g. serving as a developer: many team members play only development roles (developer, tester, analyst) and do not take on these organizational roles. The roles are not restricted to Agile coaches, but may be played by developers and business analysts. The roles we identified are the same across both participant cultures, as we demonstrate by quotes from participants from both countries below.

4.1 Mentor

Guides and supports the team initially, helps them become confident in their use of Agile methods, and encourages continued adherence to Agile practices.

Software development teams initially find Agile practices very different to their previous non-Agile work experiences. Mentoring a team in the initial stages of transitioning into Agile is extremely important. Teams may perceive the Agile practices to be simple enough to comprehend, but when it comes to actually implementing them on a daily basis, they need guidance and support which comes in the form of the *Mentor*:

> "*It's more important that you get everything right at the start. Because the process itself is not that complicated [but] doing things along the lines of the process is a little bit harder than the process itself...So with [the Mentor] it was kind of to teach us how Agile works and shape our mindset and make sure everyone knows how to work under the Agile umbrella.*" — P10, Developer, NZ

Sometimes, a *Mentor* steps in to remove misconceptions about Agile among team members. As one of the *Mentors* disclosed:

> "*We were establishing from the start and...It's mainly been showing people through that process...It's a matter of overcoming and explaining the misconceptions.*" — P2, Coach, NZ

The initial stages of transitioning into becoming a self-organizing Agile team can be very difficult. Many participants described the transitioning phase as '*a war,*' '*challenge,*' '*struggle,*' and '*difficult*' (P12, P15, P16, P18, P20). During the initial stages of transitioning, the team's existing work environment and practices must be changed to become Agile. Often the team members perceive this change as a criticism of their personal skills and retreat into a defensive corner, shunning the changes brought on by the introduction of Agile methods. A *Mentor* is quick to identify these insecurities among team members and proactively tries to clear the air of negativity from the team by encouraging them to focus on the re-evaluation of their work environment instead of their own personal skills:

> "*All the dirty doings get exposed. Hand holding people at that time... trying to take away the finger pointing and let people understand that what is being critiqued is their environment and not their working... People go into defensive mode... that's when whole negativity comes in and all Agile practices are thrown out to the wind!... [encourage] focusing on what essential good practices, fundamental framework which has to be put in place.*" — P18, Agile coach, India

The role of the *Mentor* also includes getting the team confident in their use of Agile practices by getting them used to success, one iteration after another. The *Mentor* realizes the importance of positive reinforcement and instills a sense of achievement in the team by guiding them towards successfully completing iterations and gaining positive feedback from the client:

> "*When you get the team used to success, that's where a change happens in them. You'll have a team that starts...they haven't done this before, they don't quite know how to do it. You need to show them...that they have achieved something, that they had a client presentation and the software worked...And with the next iteration...they get a little bit more confidence...And after a few such validation cycles, then they start to get confident.*" — P1, Coach, NZ

Several participants noted that members of the emerging self-organizing team face the danger of becoming relaxed and reverting to their old non-Agile ways of working (P6, P13, P22, P18). An important aspect of the *Mentor* role is to highlight the importance of continued adherence to Agile practices. The following quote describes a project where the *Mentor* was let go after the management perceived the team to be self-organizing and no longer in need of support. This turned out to be a huge mistake. In the absence of a *Mentor*, the team lost the importance of retrospectives.

> "*In the [retrospective] that we do they are so much quicker now than it used to be when we had [the Mentor] with us...[the Mentor] didn't have a vested interest in the product, she had a vested interest in the team...And now it is almost like lip service...we don't do self-evaluation as well as we used to.*" — P8, Tester, NZ

As the tester pointed out in the above quotation, the *Mentor* has a vested interest in their Agile team, as opposed to the product the team is building. It is important to maintain the *Mentor* role to support an Agile team.

The role of the *Mentor* is usually taken up by experienced Agile coaches, who display a firm understanding of both Agile methods and their teams' issues. Most *Mentors* were available to their teams on a full time basis.

4.2 Co-ordinator

Acts as a representative of the self-organizing Agile team to co-ordinate communication and change requests from customers.

We found that while the common conception that Agile teams are self-managing was largely true. There is a need for a single representative for the team to coordinate between the team and the customer representatives. In one of the Indian projects, the *Co-ordinator* role was played by a developer who helped coordinate across geographic and time boundaries with off-shored customers:

> "*Initially we avoided [having team leads]...but sometimes, because we are working offshore [it is] good to have one person who can communicate. Not a team lead in the sense not telling people what to do [but] more like co-ordinator — talks to everybody.*" — P13, Senior Management, India

The *Co-ordinator* interacts with the team on a regular and intimate level and co-ordinates communication between the team and the customers:

> "*We assign a customer representative who interacts with the team...but then passes on the feedback from the customer to the team and vice versa.*" P24, Coach, India

In case of New Zealand teams, the business analyst on the team acted as the *Co-ordinator*, representing the team to their customers and co-ordinating communication efforts.

> "*...it makes sense to have a [Co-ordinator] in the middle...if you have some sort of problem, you don't have five people asking the same question at the other end; which normally business people don't like...so having [the Business Analyst] as a [the Co-ordinator], it's working for us.*" — P10, Developer, NZ

We observed that the *Co-ordinator* also helps co-ordinate change requests made by the customers. Responding to change [17, 27] is an integral part of Agile methods and a *Co-ordinator* helps in dealing with changes in a systematic way, so that the team can respond to them effectively:

> "*[the Co-ordinator] still needs to get all the requirements to us, so whenever the business owner wants to make a change...we can plan a little bit ahead; [The Co-ordinator] might say 'OK guys, this might come in the next couple of sprints, think about it and figure out how to handle it'. So that's kind of cool.*" — P10, Developer, NZ

The team needs a clear list of requirements (Scrum's product backlog) prioritized by the customer representative before they can begin their development iteration. The *Co-ordinator* also clarifies the priority of features from customers.

> "*If [the Co-ordinator is] not there things sort of stop spinning. A lot of the time we have to come back to him: 'Is this important? Is this prioritised?...when the client says 'Oh, that's all priority' we have to go back and say 'Which?! What do you mean?!' So then [the Co-ordinator] has to go back and say 'you can't have all priority!'*" — P9, Developer, NZ

The *Co-ordinator* role was played by a developer that interfaced with off-shored customers for an Indian team and by a business analyst facing the customers as a team representative for a New Zealand team, despite the presence of the Agile coach who continued to play the *Mentor* role.

4.3 Translator

Understands and translates between the business language used by customers and the technical terminology used by the team, in an effort to improve communication between the two.

The next role that supports self-organizing Agile teams is the *Translator*. Most of our participants observed that there is a huge gap in the technical terminology that the development team is used to and the business language that the customers use. They found the need for translation between the two languages in order to ensure proper communication of product requirements from the customer representatives and clarification of issues from the development team side. The *Translator* role emerged to address and resolve this issue.

> "*(Laughs) The client always expects that the information they sent to the development team will be enough... We have meetings with them and obviously there are some gaps in the language and in the jargon... I think... technical language is a problem for business people obviously.*" — P4, Developer, NZ

Self-organizing Agile teams are responsible for collaborating effectively and frequently with customer representatives to elicit product requirements. In Scrum and XP, the customer uses story cards to specify product features in their own non-technical language, driven by business value. These story cards are meant to be broken down into tasks by the Agile team using their own technical terminology. Translating the story cards written in business language into technical specifications for development is not easy, and can lead to loss of information, as one of our New Zealand participants recounted:

"*The biggest issues with the development team ...the translation of what the client wants into something the [development team can] create. So you have a story card with some features on it. Okay, so how to turn that story card into part of a website? Development have their own interpretations of what is that they are supposed to be doing...[The] 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!'*" — P12, Senior Management/Coach, NZ

" *If you give people information with a technical bent, and they're business people, they switch off... Everything... has to be described in terms of business value... sometimes it feels hard or unnatural for a technical person to have to go to that length, and that's where skills as an analyst come in.*" — P11, Senior Management, NZ

The above observations were confirmed by a customer representative (a Scrum Product Owner) in New Zealand, who expressed their need for a *Translator* to explain the Agile development team's actions.

"*I was really struggling with [Agile vendor] and what they were doing, my knowledge isn't as technical, and really needed somebody...to give me some insight into what they were doing...they [Agile vendor] are very smart developers...but not really putting themselves in the user's shoes or the client's shoes, so some things fell through the cracks because we didn't have a good [Translator]*" — P6, Product Owner, NZ

The role of the *Translator* was also evident in translating business requirements for the Agile development teams used to technical specifications. In order to ensure the effective running of the team, this gap in languages must be addressed and overcome. The *Translator* understands the business concerns and translates them into clear and granular prioritized items for the Agile team:

"*They [customer representative] can talk about high level...'I want to have Taj Mahal', but of granite [or] marble? Even they do not have time to talk about that! This is most challenging part...getting the clarity.*" — P15, Scrum Master, India

Along the same lines, a *Translator* is able to understand the implications of change requests on business processes and effectively communicate that to the developers:

"*The terminology that developers are using and customers are using, is very different...I think the [Translator] has some value in the project. They [the organization] have a very complicated business process...some people say 'we need to change this validation rule of such and such' but they don't know the implication of those changes, it could break the whole system. And the [Translator]...understand[s] what the implication would be from these changes.*" — P5, Developer, NZ

Finally, a *Translator* can also act as a representative for team members with introverted personalities, who may be efficient in their use of technological specifications, but are unable to translate them effectively for the business customer to understand:

"*I don't normally talk to people. We ask [Translator] and then [Translator] talks to everybody else...If [customers] never knew I existed it would be great for me (laughs)...I might explain something in a very cryptic, technological way and they [customers] won't understand a word and we'd need [Translator] anyway to translate it, to make it mean something.*" — P9, Developer, NZ

The *Translator* role was played by a business analyst. The *Translator* functioned as a two-way street for translating between the technical language of the Agile teams and the business langusage of the customers. This sort of translation is imperative for the proper communication of product requirements from the customer side and clarification of issues from the development team side.

4.4 Champion

Champions the Agile cause with the senior management within their organization in order to gain support for the self-organizing Agile team.

It became evident from the interviews and observations that a self-organizing Agile team cannot emerge and flourish in isolation. The team is impacted in several ways by the environment that surrounds it. The environment that the team works and interacts in is comprised firstly of their own organization and secondly of their customers. The *Champion* is responsible for making the organizational environment conducive for the self-organizing Agile team by seeking senior management support. Lack of understanding of Agile principles and practices can lead the management to take project decisions that may damage the team. An Agile coach and trainer working with Indian Agile companies highlighted the importance of senior management support in adopting Agile at the organization:

"*The mindset change doesn't happen from the ground up and so the organizations I see getting the most benefit from Scrum, from Agile, are organizations where senior management really gets it! Where senior management has been has been through training...Senior management took the time to read, learn about Agile. The least successful Agile adoptions are ones where senior management has no interest in Agile, they have no interest in what Agile is.*" — P22, Scrum Trainer, India

Several participants highlighted the need for senior management support in attempting to establish Agile as a software development methodology within the organization (P1, P7, P12, P13, P15, P19, P21-P24). Participants observed the existence of a *Champion* on their projects, who was advocating the Agile cause within their organization to the senior management. Senior management's approval was critical to the decision to explore Agile methods within their organization andestablishing a self-organizing Agile team. The *Champion* explained the Agile values, principles, and practices to senior management and gained their support in establishing a pilot team to try Agile:

"*I [the Champion] came in with my background in Agile...explained the benefits, explained the process. The founders of the company had heard of Agile...They were keen to try it out, have somebody who could lead them...and as of four months ago all of our projects run with Agile, so the transition is complete.*" — P12, Senior Management/Coach, NZ

Gaining the support of senior management is a crucial step in being able to establish and nurture a self-organizing Agile team. An Agile coach recounted their role as a Champion and described how they convinced the senior management to explore Agile methods:

> "*I've been involved within our company to promote this notion [of Agile]...we finally got the okay...to go ahead and make it all formal. Which is excellent, but it took a [very] long time to understand people's motivations and awareness of things...talk to the business in terms that matter to them...Is it useless for them to continue with this [non-Agile] approach?*" — P1, Coach, NZ

In order to gain senior management support for exploring Agile methods, the *Champion* often resorts to setting up a pilot team to work on a project. The idea is to show to senior management how Agile practices work in reality on a small scale, thereby exhibiting the advantages of Agile software development. Some *Champions* suggested that piloting with a team that is open to trying Agile was preferred. Most *Champions* also mentioned that the initial pilot attempt works best on a project that had previously experienced difficulties with the traditional development approach, so that the value brought in by Agile was more apparent:

> "*Piloting is the key. Pilot with people who want to do it... with a project which has had problems, with changing requirements, with customers not happy. Then you'll see maximum value... if it is a hundred people organization with ten projects, try with one or two [projects].*" — P20, Developer, India

The idea of promoting Agile software development is tightly coupled with the effort of establishing an initial team. We found that promoting and gaining support for Agile as a method is important to support self-organizing Agile teams.

The role of the *Champion* is not limited to driving initial pilot projects. The *Champion* also promotes the idea of propagating more self-organizing Agile teams across the organization:

> "*The [Champion] was pretty much championing the whole Agile idea. They were thinking of using [the Champion] to expand Agile through all of [organization], so every single project they were looking at trying to put an Agile aspect to it and [the Champion] was doing all the ideas, all the objective identification, everything*" — P7, Business Analyst, NZ

4.5 Promoter

Promotes Agile with customers, and attempts to secure their involvement and collaboration to support the efficient functioning of the self-organizing Agile team.

The environmental factors that influence self-organizing Agile teams include customer collaboration and involvement, besides senior management support. Almost all participants mentioned that securing customer involvement was a challenge (P2, P4, P6, P7, P8, P9, P11, P13-17, P19-22, P24).

> "*Commitment for that time from the business [is] a huge cost...And that's something that isn't normally there in a [non-Agile] software development project [where the customers] throw [the product requirement] over the wall and don't have to worry about it for 6 months and then it appears and then you get the heartache of a product that's not working or not doing all that you want.*" — P2, Coach, NZ

> "*customers [show] reluctance to participate in sprint meetings...got no time. [There is] resistance to change [and] involvement.*" — P21, Senior Management, India

The collaboration between the team and customers ensures the development of a product that is built to the customer's vision. Convincing the customer that this advantage is worth their time investment and securing their collaboration is challenging [28]. Customers may be skeptic about their involvement in an Agile project:

> "*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.*" — P22, Scrum Trainer, India

The *Promoter* identifies the concerns of the customers and systematically attempts to engage the customer representatives by guiding them on Agile practices:

> "*I did persuade the client to go down this road... story cards, iterations, all the way through. Slowly the client did come around and started to see benefit, so it did work out really well*" — P12, Senior Management / Coach, NZ

The *Promoter* role educates the customers in Agile methods, and clears any preconceived misconceptions that they may have harboured:

> "*[Customers] would have read about [Agile] or have the wrong idea of Agile. We [coaches] have interactions with them, have a series of talks...explain to them what Agile is...[there is a] huge hullabaloo about Agile this, Agile that! We showcase our [Agile] offering - not lulling them into a sense of security but - real values and also focusing on the hardship which comes with that.*" — P18, Agile coach, India

Customer collaboration is crucial for the implementation of several Agile practices [31]. Customer involvement in the project helps the team to avoid rework:

> "*To get the client involved in the process I think is the most difficult part of Agile...[customer involvement is a] benefit for us [team], because we don't have to redo things. So from my perspective as a Developer, yes, the more the client is involved, the better for us.*" — P4, Developer, NZ

We discovered that in absence of the customer understanding of Agile and their willingness to collaborate with the team, the self-organizing team is unable to function to its full potential.

> "*Two of the [internal customers] responded lots and were very... complaining, and at the end of the project their business units loved it and the business unit 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!*" — P3, Developer, NZ

One of the developers expressed their frustration over the lack of enough customer involvement and the negative impact it had on the team's productivity:

"*It's not that we don't have the capacity, but...the business is holding off...and you know with Agile if you don't have a requirement...you can't do anything because you're supposed to be in line with business.* " — P10, Developer, NZ

It therefore becomes imperative that someone takes on the role of educating and convincing the customers to invest time and involvement in the Agile project and to collaborate efficiently with the self-organizing Agile team. It emerged that the *Promoter* took on this responsibility.

"*That is generally a problem when customer is not convinced enough to try Agile but you've got to manage both of them [Agile and customer practices]*" — P14, Developer, India

Given the collaboration-intensive nature of Agile practices, it can be dangerous to assume that a self-organizing Agile team can work and flourish in isolation. We found that the *Champion* and *Promoter* roles were crucial in identifying the influence of the environmental factors: support of senior management and customer involvement respectively. These roles were usually played by the Scrum Masters and coaches, embodied by the same person in some cases.

4.6 Terminator

Identifies team members threatening the proper functioning and productivity of the self-organizing Agile team and engages senior management support in removing such members from the team.

The role of the *Terminator* is certainly not an easy one, and perhaps the most controversial. The *Terminator* identifies individuals in team that may be hampering team productivity because of their personal characteristics and practices. Individual personality of team members can be considered more important that skill set when selecting an Agile team, as illustrated by the following reflection of a Developer:

"*[At the time of hiring] it was just 'well who is going to work better with this group of people?' rather than who's better technically or anything...personality [is] more important...I think that's really important with Agile - you've got to have people you can work that closely with and trust, a lot more than if you're doing [non-Agile].*" — P3, Developer, NZ

Many participants agreed that self-organizing Agile teams should be '*Open*' in nature and willing to '*Change*' (P1, P6, P7, P8, P11, P13, P14, P15, P16, P19, P24). In the absence of these desired characteristics, the individual is perceived to pose a threat to the proper functioning and productivity of the self-organizing Agile team. The *Terminator* identifies such individuals and engages senior management support in removing them from the team.

As one of the *Terminator* acknowledges below, the individual themselves are not '*bad*', but that they are unable to adjust to the Agile way of working and that starts to hamper the productivity of the entire team. They also note that removing such members, who hamper the self-organizing Agile team productivity, can be crucial to project success:

"*...having the right people on that team. If you have someone who isn't willing to learn and communicate - all those kind of key things that are needed in an Agile team member...any one of them that can't adjust to the Agile mechanism really needs to be removed pretty quickly...It's not that the person is bad, they may be very very good at their job, it's just that they can't adjust to the different mechanism [Agile].*" — P2, Coach, NZ

While inability to adjust into the Agile way of working is seen as a disadvantage by many *Terminators*, the other extreme of embodying idealistic or evangelist attitude towards Agile software development is also seen as a potential hindrance to the self-organizing Agile team:

"*Some evangelists have such hundred percent concepts — just scares me as a Coach... Throw out evangelists sometimes, hard reality! People get fired. It's the cold-hearted nature of this businesses, [Agile] identifies the good things, [Agile] identifies even the bad things. Sometimes [we] have to throw people out.*" — P18, Agile coach, India

Sometimes a team member can destabilize the team by their actions and even though the other team members are aware of it, they are unable to verbalize their concerns. It the *Terminator* who identifies the unverbalized concerns of the rest of the team and engages senior management support to remove the destabilizing element.

"*[Everything] seemed to go all right until [team member] tore the whole product apart...So our [Terminator] came in...noted that [team member] was holding the team back, and made an executive decision by talking to management as the [Terminator] and said 'the Agile method isn't working in this team because this one person is making such a large difference to everyone's productivity'...[we] simply didn't want to voice our opinions because there was too much fallback when we tried to...But the [Terminator] really made that quite obvious to management and therefore we [the organization] just removed them.*" — P7, Business Analyst, NZ

The *Terminator* is able to identify the individuals unable to fit into the Agile way of working and has enough senior management support to ensure their removal from the team. This role was played by experienced Scrum Masters and Agile coaches.

5. DISCUSSION

Our cross-cultural study looked at Agile practitioners from New Zealand and India. The New Zealand culture, despite being individualistic [6], did not negatively affect collaboration and coordination on these Agile teams. This is in contrast to the findings of Moe et al. who discovered that high individual autonomy proved to be a barrier to self-organization in their single project case study [35]. On the other hand, the Indian culture is hierarchical [6, 44, 46] with a low individualism (IDV) score and high Power Distance Index (PDI) [3] where managers are expected to make all decisions, a characteristic that runs contrary to the philosophy of self-organizing teams. We identified the six roles in both cultures, and the practice of the roles were unaffected by these cultural influences.

Software development teams benefit from the initial guidance of a full-time *Mentor*, played by an experienced Agile coach. The *Mentor* role is the closest to the classic Agile coach described in the Agile literature [7, 34, 38, 40]. In contrast, we found that developers and business analysts often played the *Co-ordinator* role,

despite the presence of an Agile coach on the team who continued to play the *Mentor* role.

Some studies have described individuals supporting customers by translating *technical* language to *business* language [30, 32]. A difference is that the *Translator* interacted directly with both parties and was a part of the development team. The *Translator* role was played by Business Analyst exclusively.

Both the *Translator* and *Co-ordinator* roles interact with the team on one side and the customers on the other. The *Translator* role is distinct from the *Coordinator* role in that the *Translator*'s role involves more than just passing information, but also involves translating ideas in expressions that the business/customer representatives understand into terminology that the development team is familiar with. In fact, the *Translator* can be seen as a specialization or subset of the *Coordinator* role. They were, in some cases, played by the same person.

Self-organizing teams do not emerge and flourish in isolation and are dependent on environmental factors, such as the support of senior management and the level of customer involvement. Moe et al. [35] identify lack of support system as a barrier to self-organization. Beck [5] notes that an Agile team is not equipped to handle the "*foreign relations*" with the rest of the organization by themselves. The *Champion* and *Promoter* roles handled these relationships and were mostly played by Agile coaches.

Finally, the *Terminator* role was played by Scrum Masters and coaches with the support of senior management. Cockburn and Highsmith [15] recommend placing "*more emphasis on people factors in the project: amicability, talent, skill, and communication.*" The *Terminator* exercises their power when team members did not fit in with the rest of the team, and hampered their productivity due to lack of openness and willingness to change.

We found that while one person — such as an agile coach — may play the Mentor, Champion, Promoter and Terminator roles simultaneously, each self-organisational role was only ever played by one team member at any given time. This is in contrast to development roles: all the teams we studied contained multiple developers working simultaneously.

6. LIMITATIONS

Because Grounded Theory is a very specific research method, we do not claim our results to be universally applicable: rather, they accurately characterize the contexts studied [4]. Those contexts were dictated by our choice of research destinations, which in turn were in some ways limited by our access to them.

Data derived from interviews is known to be prone to bias. We attempted to overcome this potential threat to validity using three strategies. Firstly, we observed the workplaces and activities as much as possible to supplement the data collected from interviews. The data derived from observations did not contradict, but rather supported the interview data, thereby strengthening it. Secondly, even though we focused on the development team (developer, tester, Agile coach, business analyst), we gathered a rounded perspective of the issues by interviewing practitioners representing other aspects of software development such as customer representative and senior management. Lastly, 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.

7. CONCLUSION

In this paper we set out to answer the question: *How do self-organizing Agile teams organize themselves?* We found that team members adopt one or more of 6 informal roles to facilitate their team's self-organisation.

These roles are:

1. *Mentor* that provides initial guidance, understanding, confidence of Agile methods, and encourages continued adherence to Agile practices;
2. *Co-ordinator* that co-ordinates communication and change requests from customers;
3. *Translator* that translates the customers' business language into technical terminology used by the team, and vice-versa, in order to improve communication;
4. *Champion* that gains the support of senior management to establish pilot teams and to propagate more self-organizing teams across the organization;
5. *Promoter* that secures customer collaboration and involvement to support efficient functioning of Agile teams; and
6. *Terminator* that removes team members that hamper team productivity due to their inability to fit into the Agile way of working.

All of the six roles were found in at least one project in both New Zealand and India. While some of these roles were played by Agile coaches, some others were played by developers and business analysts — the *Translator* role, for example, was played exclusively by business analysts. We hope that understanding these roles will help Agile software development teams to organize themselves, and guide Agile coaches mentoring self-organizing teams.

8. ACKOWLEDGMENTS

Our thanks to all those software practitioners who have participated in our research. This research is supported by an Agile Alliance academic grant and a NZ BuildIT PhD scholarship. Thanks to Dr. Philippe Kruchten and Dr. George Allan for their help.

9. REFERENCES

[1] N. Abbas et al. Historical roots of agile methods: Where did "agile thinking" come from? In *XP2008*, 94–103, Springer, Limerick, 2008.

[2] G. Allan. The Use of Grounded Theory as a Research Method: warts & all. In *European Conf. on Research Methodology for Business and Management Studies*, 9-19, 2005.

[3] L. R. Abraham. Cultural differences in software engineering. In *ISEC '09:*, pages 95–100, ACM, New York, 2009.

[4] S. Adolph, W. Hall, and P. Kruchten. A methodological leg to stand on: lessons learned using grounded theory to study software development. In *CASCON '08:* 166–178, ACM, New York, 2008.

[5] L. Anderson et al. Agile management - an oxymoron?: who needs managers anyway? In *OOPSLA Comp.:* 275–277, ACM, 2003.

[6] J. Aston, L. Laroche, and G. Meszaros. Cowboys and Indians: Impacts of cultural diversity on agile teams. In *AGILE '08:* 423–428, IEEE Computer Society, Washington, 2008

[7] S. Augustine, B. Payne, F. Sencindiver, and S. Woodcock. Agile project management: steering from the edges. *Commun. ACM*, 48(12):85–89, 2005.

[8] K. Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, USA, 1999.

[9] A. Begel and N. Nagappan. Usage and perceptions of agile software development in an industrial context: An exploratory study. In *ESEM '07:* 255–264, IEEE Computer Society, Washington, 2007

[10] F. Brooks Jr. *The Mythical Man-Month: Essays on software engineering.* Addison-Wesley, 1975.

[11] T. Chau and F. Maurer. Knowledge sharing in agile software teams. *Logic versus approximation:* LNCS 3075:173–183, 2004.

[12] T. Chow and D. Cao. A survey study of critical success factors in agile software projects. *J. Syst. Softw.*, 81(6):961–971, 2008.

[13] A. Cockburn. *People and Methodologies in Software Development*. PhD thesis, University of Oslo, Norway, 2003.

[14] A. Cockburn. *Crystal clear: a human-powered methodology for small teams*. Addison-Wesley Professional, 2004.

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

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

[17] T. Dybå and T. Dingsoyr. Empirical studies of Agile software development: A systematic review. *Inf. Softw. Technol.*, 50(9-10):833–859, 2008.

[18] S. Fraser et al. Xtreme programming and Agile coaching. In *OOPSLA Comp. '03:*, 265–267, ACM, New York, 2003.

[19] P. Fitsilis. Comparing PMBOK and Agile Project Management software development processes. SCSS(1), 378-383, 2007.

[20] B. Fitzgerald, G. Hartnett, and K. Conboy, Customising agile methods to software practices at Intel Shannon. *Eur. J. Inf. Syst.*, 15(2): 200-213, 2006.

[21] B. Glaser. *Theoretical Sensitivity*. Sociology Press, Mill Valley, California, 1978.

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

[23] B. Glaser and A. L. Strauss. *The Discovery of Grounded Theory*. Aldine, Chicago, 1967.

[24] 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.

[25] J. A. Highsmith, III. *Adaptive software development: a collaborative approach to managing complex systems*. Dorset House Publishing, New York, 2000.

[26] J. Highsmith. *Agile Project Management: Creating Innovative Products*. Addison Wesley, USA, 2004.

[27] J. Highsmith and M. Fowler. The Agile Manifesto. *Software Development Magazine*, 9(8):29–30, 2001.

[28] R. Hoda, J. Noble, and S. Marshall. Agile Undercover: when customers don't collaborate (To appear) In *XP2010*, Norway, 2010.

[29] PMI Institute A Guide to the Project Management Body of Knowledge. PMI Standard Committee, 2004.

[30] Mann, C. and Maurer, F. A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction. In *ADC*, IEEE Computer Society, 70–79, USA, 2005.

[31] A. Martin, R. Biddle, and J. Noble. The XP customer role in practice: Three studies. In *ADC '04:*42–54, IEEE Computer Society, Washington, 2004

[32] A. Martin, R. Biddle, and J. Noble. The XP customer role: A Grounded Theory. In *AGILE2009*, IEEE Computer Society, Chicago, 2009.

[33] R. Martin. *Agile Software Development: principles, patterns, and practices.* Pearson Education, NJ, 2002.

[34] N. B. Moe and T. Dingsoyr. Scrum and team effectiveness: Theory and practice. In *XP2008*, 11—20, Springer, Limerick, 2008.

[35] N. B. Moe, T. Dingsoyr, and T. Dybå. Understanding self-organizing teams in agile software development. In *ASWEC '08:* 76–85, IEEE Computer Society, Washington, 2008

[36] S. Nerur, R. Mahapatra, and G. Mangalaraj. Challenges of migrating to agile methodologies. *Commun. ACM*, 48(5):72–78, 2005.

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

[38] L. Rising and N. S. Janoff. The Scrum software development process for small teams. *IEEE Softw.*, 17(4):26–32, 2000.

[39] K. Schwaber *Scrum Guide*. Scrum Alliance Resources, 2009.

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

[41] H. Sharp and H. Robinson. An ethnographic study of XP practice. *Empirical Softw. Engg.*, 9(4):353–375, 2004.

[42] H. Sharp and H. Robinson. Collaboration and co-ordination in mature extreme programming teams. *Int. J. Hum.-Comput. Stud.*, 66(7):506–518, 2008.

[43] J. Stapleton. *Dynamic Systems Development Method*. Addison Wesley, 1997.

[44] M. Summers. Insights into an Agile adventure with offshore partners. In *AGILE '08:* 333–338, IEEE Computer Society, USA, 2008.

[45] H. Takeuchi and I. Nonaka. The new new product development game. *Harvard Business Review*, 1986.

[46] E. Uy and N. Ioannou. Growing and sustaining an offshore Scrum engagement. In *AGILE '08:* 345–350, IEEE Computer Society, USA, 2008.

[47] E. Whitworth and R. Biddle. The social nature of Agile teams. In *AGILE'07:* 26–36, IEEE Computer Society, USA, 2007.