Interdisciplinary Journal of Information, Knowledge, and Management

An Official Publication of the Informing Science Institute
InformingScience.org

IJIKM.org

Volume 14, 2019

AGILE SELF-SELECTING TEAMS FOSTER EXPERTISE COORDINATION

Mawarny Md Rejab*	Universiti Utara Malaysia, Sintok, Kedah, Malaysia	mawarny@uum.edu.my
James Noble	Victoria University of Wellington, Wellington, New Zealand	kjx@ecs.vuw.ac.nz
Stuart Marshall	Victoria University of Wellington, Wellington, New Zealand	stuart.marshall@vuw.ac.nz

* Corresponding author

ABSTRACT

Aim/Purpose	This paper aims to discuss the activities involved in facilitating self-selecting teams for Agile software development projects. This paper also discussed how these activities can influence the successful expertise coordination in Agile teams.
Background	Self-selecting teams enable Agile team members to choose teams based on whom they prefer to work with. Good team bonding allows Agile team members to rely on each other in coordinating their expertise resources effectively. This is the focal point where expertise coordination is needed in Agile teams.
Methodology	This study employed Grounded Theory by interviewing 48 Agile practitioners from different software organizations mainly based in New Zealand. This study also carried out several sessions of observations and document analysis in conjunction with interviews.
Contribution	This study contributes to the body of knowledge by identifying the way self-selecting teams support expertise coordination.
Findings	Our findings indicated that the activities involved tend to influence the successful expertise coordination in Agile teams. Self-selecting teams are essential to supporting expertise coordination by increasing inter-dependencies between Agile team members, ensuring a diverse range of knowledge and skills in teams.

Accepting Editor Jeffrey Alstete │ Received: November 26, 2018 │ Revised: March 20, 2019 │ Accepted: March 29, 2019.
Cite as: Md Rejab, M., Noble, J., & Marshall, S. (2019). Agile self-selecting teams foster expertise coordination. *Interdisciplinary Journal of Information, Knowledge, and Management*, *14,* 99-117. https://doi.org/10.28945/4280

Recommendations for Practitioners	The self-selecting team activities can be used as a guideline for Agile software organizations in forming self-selecting teams in the fastest and most efficient way. It is vital for management to facilitate the process of self-selecting teams in order to optimize successful expertise coordination.
Recommendations for Researchers	There is potential for further Grounded Theory research to explore more activities and strategies involved in self-selecting teams.
Impact on Society	Self-selecting teams in Agile software developments projects tend to boost the productivity of software development.
Future Research	Several hypotheses can be tested through a deductive approach in future studies.
Keywords	agile software development, self-selecting teams, expertise coordination, grounded theory

INTRODUCTION

Agile teams are expected to be self-organizing teams, starting with the initial stage of forming teams (Blankenship, 2018). However, traditional team formation is still being applied in some Agile organizations. The management of the organization decides the membership of the teams based on the expertise required for the software project.

The starting point to embrace self-organizing teams is through self-selecting teams, which enable Agile team members to form their own teams (Blankenship, 2018). This approach is consistent with self-organizing practices which are shifting from command-control management to more self-management culture.

Through Grounded Theory, we discovered three activities required in facilitating self-selecting teams for Agile software development projects: setting up cross-functional teams, selecting the right team, and balancing teams. Self-selecting teams enable Agile team members to choose teams based on whom they prefer to work with. Good team bonding allows Agile team members to rely on each other in coordinating their expertise resources effectively. This is the focal point where expertise coordination is needed in Agile teams.

Expertise coordination is defined as "the management of knowledge and skills dependencies" (Faraj & Sproull, 2000). This definition shows how team members should depend on each other in managing and utilizing their expertise resources. Expertise coordination requires a team to recognize who has particular expertise, when and where they are needed, and how to access that expertise effectively (Faraj & Sproull, 2000). Hence, locating and recognizing the source of expertise is a pivotal step in coordinating a team's expertise, as well as expertise outside Agile teams (Rejab, Noble, & Allan, 2014a, 2014b; Rejab, Noble, & Marshall, 2015).

Even though many studies emphasize coordination in Agile teams, there is a paucity of empirical studies that focus on expertise coordination. Abrahamsson, Conboy, and Wang (2009) suggest further research is needed to explore the human resource dependencies faced by Agile teams, including expertise. There is clearly a need to conceptualize and theorize the underpinnings of expertise coordination from an Agile Software Development perspective.

The aim of this study is to build a theory of expertise coordination in Agile Software Development projects, which is guided by the following research questions:

> What are management strategies support expertise coordination in Agile Software Development projects?

This question attempts to determine the roles of management in supporting expertise coordination in Agile Software Development projects. Successful expertise coordination relies on management support through self-selecting teams. Management should promote self-selecting teams, which enable Agile team members to form their own teams. The right team with the right team members can be formed through self-selection, which assists Agile team members in sharing and accessing expertise when it is needed.

This paper aims to discuss the activities involved in facilitating self-selecting teams for Agile software development projects: setting up cross-functional teams, selecting the right team, and balancing teams. We found that self-selecting teams indirectly tend to influence the successful expertise coordination in Agile teams.

The rest of this paper is structured as follows: the next section describes Grounded Theory; the third section presents the findings of this study; the fourth section discusses these findings; and the last section puts forward conclusions.

LITERATURE REVIEW

Agile Software Development is a group of software development methods that promote iterative and incremental development (Larman & Basili, 2003). Agile Software Development projects emphasize effective teamwork by concentrating on people. People are the most important factor in determining the success of Agile Software Development projects (Cockburn & Highsmith, 2001; Fowler & Highsmith, 2001). Agile methods are designed to capitalize on every individual's expertise, but to produce a quality software product, the presence of expertise is not sufficient (Faraj & Sproull, 2000); it is important to leverage the available expertise through expertise coordination.

Expertise coordination refers to how team members depend on each other to manage and utilize their expertise. Expertise coordination requires a team to recognize who has particular expertise, when and where that expertise is needed, and how to access the expertise effectively in a timely manner.

Strode and Huff (2012) categorize expertise coordination under cognitive coordination. Cognitive coordination occurs when team members perform tasks and need to understand other team members, and to adjust their behaviour in order to cope with others (Rico, Sánchez-Manzanares, Gil, & Gibson, 2008). Teamwork and collaboration are basic requirements that lead to cognitive coordination. Cognitive coordination is often known as implicit coordination, since the coordination relies on tacit knowledge without emphasizing explicit knowledge. Shared mental models (Levesque, Wilson, & Wholey, 2001) collective mind (Crowston, Annabi, Howison, & Masango, 2004), and expertise coordination (Faraj and Sproull, 2000) are classified under cognitive coordination as they are forms of implicit coordination.

Expertise coordination involves socially shared cognitive processes to enable the knowledge and skills dependencies (Faraj and Sproull, 2000). Transactive Memory Systems (TMS) provide a dynamic of human connection which fosters socially shared cognitive processes (Akgun, Byrne, Keskin, & Lynn, 2006). TMS exists when two or more people interact cooperatively in storing, retrieving, and communicating information (Wegner, 1987). Wegner (1987) conceived the term of TMS, which focuses on integrating and utilizing expertise in order to optimize the value of members' knowledge (Lewis, 2003).

Transactive Memory Systems (TMS) theories are used to coordinate knowledge, people, resources, tasks, and technology efficiently. TMS is a theory of knowledge coordination which describes interconnected individual memory systems in sharing knowledge (Wegner, 1995). TMS occurs through shared mental models, which consists of a directory of who knows what and a process of encoding and retrieving information. Akgun, Byrne, Keskin, Lynn, and Imamoglu (2005) revealed the roles of TMS in supporting expertise coordination as follows:

- Reducing the cognitive load in remembering each member's expertise.
- Accessing a pool of expertise.
- Facilitating the process of sharing tacit knowledge of different domains.
- Allocating resources according to member expertise.

Management needs to take appropriate actions to develop shared mental models. Therefore, the role of management is vital to support expertise coordination since shared mental models are essential for coordinating expertise. Management support is necessary for coordinating expertise through self-selecting teams.

Self-selection is a facilitated process that allows Agile team members to self-organize in forming cross-functional teams (Sandy & David, 2014). Self-selection empowers Agile team members to work with people that they want to work with and choose what they prefer to work on. This will lead to formation of high performance Agile teams with interdisciplinary team members. Forming a high performance team does not necessarily pick all the best manpower, however, identifying the best combination of people with the interdependent skills and expertise, preference, and personalities.

Tuckman (1965) proposed four main phases of high performing team formation: forming, storming, norming, and performing. However, Agile software development projects are not expected to employ these orderly fashion and traditional phases. Therefore, several Agile practitioners proposed processes of self-selecting teams that have been applied in their organizations (Sandy & David, 2014; Magnes, 2016). For instances, Magnes (2016) introduced four activities in self-selecting teams as follows:

- Describe what each crew will be working on
- Agree what skills are needed for each crew
- Select the crews
- Choose team members

Self-selecting team formation is challenging since many activities or workflows need to be carried out. Transition to Agile methods, however, should not neglect self-selecting team formation. Empowerment of self-selecting teams is stressful at the beginning (Colomo-Palacios, González-Carrasco, López-Cuadrado, & García-Crespo, 2012). After practicing self-selected teams for a while and gaining its benefits, Agile teams will appreciate this.

RESEARCH METHOD

This study employed Grounded Theory as a research method. Grounded Theory is widely used as a research method for developing a theory in many fields of study including software engineering (Dorairaj, Noble, & Allan, 2013; Hoda, Noble, & Marshall, 2012; Waterman, Noble, & Allan, 2015). Grounded Theory is an inductive research method that aims to infer new theories from observed data (Glaser & Strauss, 1967). There are several reasons why Grounded Theory is applicable as a research method for this study. Firstly, Grounded Theory is appropriate for exploring human behaviour and social interactions (Glaser, 1992). This study focuses on how to assess the work performance of Agile team members. Secondly, Grounded Theory is appropriate to be used in areas that are underexplored which require further investigation (Birks & Mills, 2011). Further investigation is needed to conceptualize and theorize about the underpinnings of performance appraisal for Agile teams members.

The following steps outline the general process of Grounded Theory for this study adapted from Hoda et al. (2012):

- Defining a research area
- Continuous data collection

- Continuous data analysis
- Evaluating the theory

Defining a Research Area

Conducting a minor literature review assists us as researchers to identify an area of interest. A minor literature review is conducted to ensure the emerging theory is based solely on observed data, without imposing any preconceived ideas from the literature. This study began by exploring Agile Software Development practices and its implementations. Based on the literature review, expertise coordination was identified as an area requiring further investigation and we have decided to explore this area in depth. In order to explore the research area in depth, it is possible to formulate research questions. Formulating research questions, however, is not an integral part of Grounded Theory. The research question assists researchers to determine suitable methods of collecting data and detail the emerging theory (Charmaz, 2006; Creswell, 2012).

Prior to conducting data collection, we formulated the defined broad research question that related to this study. As findings of this study were emerging, the research questions, however, have been amended or changed in order to reflect the emergent concepts.

Data Collection

Glaser and Strauss (1967) emphasize the importance of collecting data with multiple methods, which promise the construction of a novel theory. Although interviews are the primary data collection method used in Grounded Theory studies (Birks & Mills, 2011; Creswell, 2012), researchers can employ other methods. This study employed interviews as the predominant source of data collection, in conjunction with observations and document analysis.

We started to recruit more participants after we joined the meetup group for the Agile Professionals Network branch in Wellington, New Zealand. Besides that, we used Agile conferences such as XP conferences, and Agile conferences based in New Zealand and the United States, as a platform to recruit more Agile practitioners. Attending Agile workshops, which were run by Agile training companies, was another avenue to identify and recruit potential participants in this study.

Semi-structured interviews were carried out with 48 Agile practitioners from different software organizations mainly based in New Zealand as depicted in Table 1. The participants engage in different business domains such as education, finance, and human resources. This study was open to Agile practitioners who apply Agile practices in their software development projects.

Table 1. Summary of Research Participants and Agile Projects

PERSON	LOCATION	AGILE ROLE	AGILE METHOD	PROJECT DOMAIN
P1	New Zealand	Developer	XP and Scrum	Mobile application
P2	New Zealand	Agile Coach	XP, Scrum and Kanban	Not disclosed
P3	Australia	Agile consultant	Not disclosed	Not disclosed
P4	New Zealand	Agile Coach	XP and Scrum	Education
P5	New Zealand	Software Tester	Not disclosed	Printing
P6	Australia	Team Leader	Not disclosed	Accounting
P7	New Zealand	Agile Consultant	XP and Scrum	Financial
P8	Australia	Agile Coach	Scrum, XP, Kanban, Lean	Human Resources
P9	New Zealand	Business Analyst	Not disclosed	Insurance
P10	New Zealand	Software Tester	Scrum	Education

PERSON	LOCATION	AGILE ROLE	AGILE METHOD	PROJECT DOMAIN
P11	New Zealand	Project manager	Scrum	Education
P12	New Zealand	Agile Coach	Scrum and Kanban	Not disclosed
P13	New Zealand	Agile Coach	Scrum and Kanban	Government application
P14	New Zealand	Product Owner	Not disclosed	Not disclosed
P15	New Zealand	Agile Coach	Scrum and Kanban	Government application
P16	New Zealand	Agile Coach	Scrum and Kanban	Goverment application
P17	New Zealand	Software Tester	Scrum	Education
P18	New Zealand	Developer	Scrum	Education
P19	New Zealand	Business Analyst	Scrum	Education
P20	New Zealand	User Experience Designer	Scrum	Not disclosed
P21	New Zealand	Agile Coach	Scrum and Kanban	Education
P22	New Zealand	Scrum Master	Scrum, kanban and XP	Education
P23	New Zealand	Developer	Scrum and XP	Dataware house
P24	New Zealand	Scrum Master	Scrum and Kanban	Banking
P25	New Zealand	Developer	Scrum and Kanban	Financial
P26	New Zealand	Team Leader	Scrum and XP	Government Application
P27	New Zealand	Developer	Scrum and XP	Fishery
P28	Sweden	Developer	Kanban	Telecommunication
P29	Denmark	Developer	Scrum	Medical
P30	India	Business Analyst	Scrum and Kanban	Not disclosed
P31	Malaysia	Scrum Master	Scrum and Kanban	Broadcast
P32	Malaysia	Scrum Master	Scrum, Kanban and XP	Enterprise
P33	Malaysia	Project Manager	Scrum and Kanban	Security Application
P34	United States	Agile Coach	Scrum	Financial
P35	United States	Developer	Scrum	Financial
P36	United States	Developer	Scrum	E-commerce
P37	United States	DevOps	Not disclosed	Not disclosed
P38	United States	User Experience Designer	Not disclosed	Not disclosed
P39	United States	Agile Coach	Scrum and XP	Not disclosed
P40	United States	Stakeholder	Not disclosed	Not disclosed
P41	United States	Agile Coach	Scrum and XP	Biotechnology
P42	United States	Tester	Scrum and XP	Retail
P43	Wellington	DevOps	Scrum and XP	Not disclosed
P44	Wellington	Architect	Scrum	Retail
P45	Wellington	Tester	Scrum and Kanban	Financial
P46	Wellington	Agile Coach	Scrum, Kanban and XP	E-commerce

PERSON	LOCATION	AGILE ROLE	AGILE METHOD	PROJECT DOMAIN
P47	Wellington	Developer	Scrum, Kanban and XP	E-commerce
P48	Wellington	Tester	Scrum and Kanban	Banking and Media

Theoretical sampling is a way to ensure the validity of a study by selecting subsequent participants for data collection based on existing data analysis (Glaser & Strauss, 1967). Theoretical sampling should ensure that other perspectives are gained from the identified participants and drawn indirectly from a broad range of other participants. This study considered a broad range of Agile roles including external specialists in order to enable the triangulation of findings. Different roles provide different insights and perspectives toward external expertise coordination. We interviewed Agile team members as well as external specialists such as User Experience Designers, Software Architects, and DevOps (Development and Operation). We stopped the data collection once we reached theoretical saturation, i.e., when no new data emerged (Glaser & Strauss, 1967).

In Grounded Theory, there is no fixed sample size in recruiting participants (Baker, Edwards, & Doidge, 2012). The researcher continues to collect data until theoretical saturation has been reached, when no new data emerges (Glaser & Strauss, 1967). Therefore, the number of participants depends on the emergent findings. In order to have wide coverage of phenomena studied, approximately 30 to 40 interviews are generally adequate to reach theoretical saturation (Stern & Porr, 2010). We recruited 48 Agile practitioners, which is significantly more than the typical number of participants.

This study employed observations and document analysis in conjunction with interviews. Observations and document analysis are classified as secondary data collection methods of this study. The main purpose of these methods is to confirm the accuracy of interviews data and enhance the validity of that data (Kirk & Miller, 1986). Observations provided a great opportunity to view the actual participants' behaviour when they were engaging in Agile Software Development. Moreover, observations allowed us to gain a deeper understanding of the participants' settings (Kolb, 2012; Parry, 1998). The advantages of observations led us to identify new findings and also enabled data triangulation. Thus, we have carried out six observational sessions over two months, observing two different Agile teams, who worked on different software development projects. Table 2 depicts observation details at company XYZ based in Wellington, New Zealand.

Table 2. Observation Deatils at company XYZ

TEAM	PROJECT DOMAIN	TEAM SIZE	PROJECT DURATION
A	Metadata search tool and various	4 developers 1 scrum master	5 years
B	New corporate website	2 developers, 1 designer 1 scrum master	12 weeks

Document analysis is another alternative for us to collect and elicit more data (Charmaz, 2006; Glaser & Strauss, 1967). Document analysis involves reviewing or evaluating printed or electronic documents, that have been produced without researchers' intervention (Bowen, 2009). We collected relevant documents during interviews and observation, including a sample of a form used for forming self-selecting teams.

During analysis, we looked for consistency between interviews, documents, and observations. Sometimes, document analysis sparked new ideas that led to further data collection and the emergence of new findings.

Data Analysis

Data analysis begins as soon as the first interview has been conducted and continues until the emergence of a core category (Corbin & Strauss, 1990). In this study, we employed three stages of coding proposed by Glaser and Strauss (1967): open coding, selective coding, and theoretical coding. Open coding and selective coding intend to produce substantive codes. Substantive codes are the emergent categories and properties that conceptualize the phenomena studied (Glaser, 1978). The purpose of theoretical coding is to generate theoretical codes, which conceptualize interrelation between substantive codes as hypotheses or emergent theory (Glaser, 1978).

Open coding involves a process of examining the data, breaking up the data into segments and then attaching a label for each segment (Stern and Porr, 2010). As line-by-line coding is a time-consuming and painstaking effort (Glaser, 1978), we decided to use key point coding. We examined words, phrases, and sentences in order to collate key points (Allan, 2003). Then we rephrased the key point with a meaningful label, which led to the formation of a code. This step of coding excited the researchers because digging out new codes led to rich research findings.

In order to look for commonalities and differences, we used constant comparison to compare codes and codes, codes and concepts, and concepts and concepts (Glaser and Holton, 2004). Constant comparison is a central process of Grounded Theory, which purposely intends to identify patterns and variations of codes, concepts, and categories (Charmaz, 2006; Strauss & Corbin, 1998). Through constant comparison, existing concepts, and categories can be enhanced or new concepts or categories formed (Urquhart, Lehmann, & Myers, 2009). We first used constant comparison to compare codes with one another for establishing uniformity between codes. Similar codes with common themes were grouped together and emerged as a concept.

Many concepts emerge, and constant comparison is repeated until concepts form a category. A category is a group of similar concepts that is used to generate a theory. Several categories have emerged from our data analysis such as locating and recognizing expertise (Rejab et al., 2014b), distributing expertise (Rejab et al., 2014a), and coordinating outside expertise (Rejab et al., 2015). This paper presents the category *self-selecting teams* which is discussed in the next section.

As we proceeded with constant comparison, a core category started to emerge. A core category is the highest level of data abstraction in Grounded Theory, which represents the main theme or concern of participants (Glaser, 1978). Glaser (1978) argues that the core category must be central and related to many other categories and their properties. The core category is selected based on its frequent reoccurrence during coding.

We moved to the next step, selective coding, after the core category has emerged (Glaser, 1978; Glaser 1992). At this stage, the core category acts as a baseline for further data collection and analysis, as well as theoretical sampling. Then, we analysed data based on the emergent core category, which was coded selectively on categories that were related to the core category.

Theoretical coding is the final stage of coding, after theoretical saturation. Theoretical coding generates theoretical codes, which conceptualize how substantive codes are related to each other as hypotheses, and then are integrated into a theory.

Evaluating a Theory

This study employed multiple approaches for evaluating the emergent theory:

- Lincoln and Guba's criteria
- Glaser's criteria
- Weber's criteria

Guba and Lincoln's criteria (Guba and Lincoln, 1994) was used as a threat to validity of this study. This study considered four criteria for evaluating our study as a qualitative study: credibility, transferability, dependability, and confirmability. Each criteria has specific approaches for ensuring the trustworthiness of this study. For instance, credibility was examined through prolonged engagement, persistent observation, triangulation of data, peer debriefing, negative case analysis, and member checks.

Specific to Grounded Theory, we evaluated our study according to Glaser's criteria (Glaser, 1998; Glaser, 1978): fit, work, relevance, and modifiability. Besides Glaser's criteria, this study has also evaluated the quality of emergent theory based on Weber's five criteria: importance, novelty, parsimony, level, and falsifiability (Weber, 2012).

RESEARCH FINDINGS

The findings of this study revealed three activities required in facilitating self-selecting teams for Agile software development projects as follows:

- Setting up cross-functional teams
- Selecting the right team
- Balancing teams

SETTING-UP CROSS-FUNCTIONAL TEAMS

Cross-functional teams include different functional expertise required to develop software. The team is set up based on a business domain area, which enables the management of the organization to set and clarify goals and objectives for each team. The management then can define the roles required for each team. Based on the findings of this study, different skills are required in a team in order to fill basic roles. The roles might change depending on requirements of the software project:

> "We have a changeable blue print. So, we think the skills are business analyst, designer, developer, tester and some kind of infrastructure production. We think all squads even though they vary, need those skills. Each squad also needs a Scrum Master and also a product owner as well as to define what we are going to build. Generally, it is about seven skills."- P46, Agile Coach.

The number of team members required for each role varies depending on software projects. Minimizing the team size, however, needs to be considered when setting up Agile teams:

> "Each squad is made of 3 to 7 people. In some cases, the biggest squad has one person per skill."- P46, Agile Coach.

> ``The number of team members is different but there is a baseline. Most teams at least have two developers and testers. Each team needs a product owner or a scrum master. It is like a basic team. On top of that, depending on the nature of the team, we might have a designer and BA, but not for all teams. It depends on work."- P47, Developer.

According to some participants (e.g., P12, P21, P22, P32, and P24) a big team causes several problems that hinder successful expertise coordination. There will be more communication and interaction lines once more team members are involved, as well as a wide range of knowledge and skills. As a larger number of team members are involved, however, it will be hard to identify who has the sort of expertise needed and how to access the required expertise:

> "We have a team which is bigger than 9 people. We see lots of problems in sharing expertise [sic]. There are many different lines of communication and it is hard to coordinate."- P21, Agile Coach.

> "In the big team, the more interactions involved and the more assumptions made, the more problems need to be fixed later on." - P24, Scrum Master.

The next major problem is the accessibility of team members. For instance, participant P22 mentioned that his team members refused to attend daily stand-up meetings. A daily stand up meeting provides a communication vehicle for Agile team members to raise issues and obstacles that impede their progress. As there are many members involved, it is hard to strictly monitor and manage every member. If they regularly miss meetings, more problems will occur including degrading expertise coordination. Failing to attend the meeting will affect the process of finding and divulging the necessary expertise and solution:

> "`They missed the stand-up meeting because it is too big." - P22, Scrum Master.
>
> "What I found in a small team is that it is easier to get everybody in the room and have everybody involved in discussion and conversation."- P24, Scrum Master.

Therefore, a small team is important for Agile team members to bond together easily and directly induces awareness of team members' expertise and performance. Participant P22 indicated that a small team provides a space for him to notice other team members who are facing problems or obstacles in accomplishing their tasks. It is important to immediately identify those people in order to provide relevant expertise in solving their problems:

> "When they are in the small team, they bond better, are dynamic and more manageable. It is also easy to see someone who is struggling for help."- P22, Scrum Master.

An interesting point that has been revealed by one of the participants is the velocity of performance. A small team tends to speed up the process of forming the team and assists team members to quickly bond together. Therefore, a small team also tends to speed up the processes of expertise coordination in Agile teams, including identifying, recognizing, and accessing relevant expertise:

> "A small team is much easier to communicate with and we get much more velocity. Forming the team is quicker and we know each other and work together better."- P24, Scrum Master.

It is important to set up Agile teams properly based on the desired goals, objectives, and roles needed for each team. Failure in setting up the right Agile teams, particularly with an optimal size, tends to affect the next steps in self-selecting Agile teams.

Selecting the Right Team

Effective self-organizing teams require the management to allow employees to choose which software project they intend to work on and whom they prefer to work with. Before selecting a team, every potential Agile team member is typically provided with an overview of the software project involved, including the goals and objectives of each team. The purpose of this overview is to ensure they clearly understand their roles and responsibilities. Based on the information given, they will consider the diversity of skills and experience of potential team members when selecting the team:

> "So, we provided information up-front, which is more details about what they will be signed up for. So, they can have some conversation before [selecting the team]. During the self-selection team, we work hard to make sure the right people were talking together. So, I and another coach will help to facilitate the conversation."- P46, Agile Coach.

The findings of this study indicated that several factors that participants (e.g., 6, 23, 46, and 47) tend to consider when selecting a team: types of work, team members, and knowledge sharing. It is common to select a team based on types of work or project involved:

> "When everyone is selecting a team, they look at what a type of work that they want to do."- P47, Developer.

Every Agile team member knows their capabilities and strength and how they can contribute for ensuring the project success. They try to match their expertise with the requirements of the project when selecting their preferred team:

> “I focus more on something that I can provide, my expertise and what type of project."- P23, Developer.

In selecting a team, some Agile team members consider who they prefer to work with. Choosing the right team members will enable them to work well together:

> “If they see someone's name and they think that they won't work well with that person, they will change to another squad."}- P47, Developer.

Participant, P47, however, argued that Agile teams put priority on work instead of team members when deciding their preferred teams.

> “Closeness to others in the organization definitely plays a part but not the main one [sic]."- P47, Developer.

Participant P46 pointed out that knowledge sharing is a pivotal factor when Agile team members select their team. They chose their preferred team based on what knowledge and skills that they can gain from the team and what they can contribute in enhancing the team knowledge:

> “There are two questions they asked themselves. One, 'what can I learn from this squad?'..... Second, is to be an expert. 'What can I teach this squad?'."- P46, Agile Coach.

In order to enable knowledge sharing, a diversity of expertise and experience are essential for each team. An Agile team should consist of different roles and sharing expertise exists between same roles, and within and across different roles as well:

> “We are not just sharing knowledge between developers, but also between developers and testers, testers and business analysts."- P6, Agile Coach.

Based on our observation at a software development company based in Wellington, New Zealand, we found active knowledge sharing happened between developers or between product owners, and between all team members as well. At this company, during team selection, team members consider how well they can work with and how well they can share knowledge together.

Besides roles composition, mixing junior and senior team members in a team is the best practice in forming Agile teams. Ensuring the diversity of expertise and experience, however, should reflect the needs and requirements of the software project:

> “So, people want to join the squad with novices, beginners or juniors, because they want an opportunity to help them and coach them."- P46, Agile Coach.
>
> “As a senior, I have to work with junior staff. Because most of the time, I'm not probably producing something for myself but I teach others."- P47, Developer.

Knowledge sharing creates a win-win situation, which allows juniors as well as seniors to gain benefits in sharing their knowledge and skills. From juniors' point of view, they have opportunities to develop their individual expertise with guidance from seniors. Besides enhancing team knowledge, seniors also have chances to be experts in their particular area of expertise and this indirectly tends to improve their coaching skills.

In term of implementation, self-selection is quite difficult for new staff who have recently joined the organization. Based on the findings of this study, they took more time than senior members to decide their preferred team. The main reason is how well they can work with others. Their curiosity towards others’ inter-personality, attitude, capability, and expertise urges them to make a decision for selecting a team:

> “If they have been around a while, they could very quickly make decision. If they have just joined this company, it is very hard to make that call. They don't know enough whether they can work well together." - P46, Agile Coach.

Therefore, selecting the right team requires facilitation from coaches and management of organization. Proper guidance from them will assist Agile team members to have a valuable space for conversation and get the right information before choosing a team. This will facilitate the right implementation of self-selecting team, which tends to form the right team with the right team members.

BALANCING TEAMS

After selecting a team, the management of the organization and the project leader have to ensure well-balanced teams are formed. Each team should be composed of the right roles and appropriate number of team members, which align with the needs and objectives of team:

> “So, everyone selects a team that they want to work for. At the end of the day, the team needs to be balanced. We can't have five developers here and one and only in the other team."- P47, Developer.

Besides roles and number of team members, balancing background experience between junior and senior members also need to be taken into account. Working with team members of same level expertise tends to hinder effective knowledge sharing. Compared to senior members, junior members tend to feel negative effects when they are left behind in developing their knowledge and skills without support and guidance. Therefore, it is important to balance levels of experiences within teams, even though there is no specific rule in deciding the composition of number of junior and senior members in a team:

> “Some squads don't want too many junior developers in the team because it requires a lots of learning."- P47, Developer.

Balancing the teams requires persuasive skills by the project leader and management of organization. The finding of this study indicated that persuasion was used to balance the teams. This provides an opportunity for Agile team members to consider the advantages and disadvantages of joining other teams. The final decision, however, is still made by Agile team members in self-selecting teams:

> “But when the team balance is not quite right, then the management and project leader will come to convince us to think about the other squad. Some people choose to do that after they convinced them to join the other team. They don't force us to move to the other squad. But, they tell us what is the best for the company."- P47, Developer.

Balancing teams needs to consider how to ensure the teams are likely to be stable. Team stability requires Agile teams to remain intact for the long term, truly gel together, progress well together, and trust each other. The findings of this study revealed that self-selecting teams have a positive impact on team stability because little fluctuation in team composition has been reported:

> “Our squads are normally stable but there is some movement. If they decide to leave the squad, they need to have a conversation with the other squad."- P46, Agile Coach.

Having stable teams can be beneficial for coordinating expertise in Agile teams. Little fluctuation in changing team composition enables the stability of skill set. Team members do not need to frequently update their meta-knowledge of the available expertise in their teams:

> “How can the team learn from the person who comes in and or the person who comes in right and disappears. If we get somebody else in, how can we extract the knowledge and learn from them. The skill set has to be sustainable [sic]."- P16, Agile Coach.

Stable teams induce shared responsibilities to ensure the quality of the whole software development. Participant P2 pointed out that tight bonding between Agile team members enables them to be more responsible toward others’ roles. They tend to absorb others’ roles when their expertise is needed:

> “They develop a good sense of maturity and run the team well and they can see roles in their team as their own."- P2, Agile Coach.

The final activity of self-selecting teams, balancing teams, assists in improving the team structure with the right roles and well-balanced levels of experienced within teams. The management of organization and project leader involvement, however, are important for balancing teams with consideration and cooperation from Agile team members in shifting to the other team.

Self-selection teams is a baseline in supporting expertise coordination in Agile teams. The right team with the right team members can be formed through self-selecting teams, which assists Agile team members in sharing and accessing expertise when it is needed. Furthermore, minimizing team size, balancing and stabilizing teams can be implemented through the sequence activities of self-selecting teams, which have a positive impact on expertise coordination. Therefore, it is vital for the management of organization to implement self-selecting teams in supporting expertise coordination.

DISCUSSION

Self-selecting teams are formed when Agile team members choose their own teams and decide whom they are intended to work with. Even though various different approaches are available to form Agile teams, Olsson, Bosch, and Alahyari (2013) and Scott and Pallock (2006) affirm that self-selecting teams are effective and suit the Agile Software Development environment. This is aligned with our findings, which indicated that self-selecting teams tend to support expertise coordination by:

- Increasing dependencies between Agile team members
- Ensuring a diverse range of knowledge and skills in teams

We found self-selecting teams increase levels of interaction between team members, which is aligned with Scott and Pallock (2006). Self-selecting teams enable Agile team members to choose teams based on whom they prefer to work with. Good team bonding allows Agile team members to rely on each other in coordinating their expertise resources effectively.

Furthermore, Scott and Pallock (2006)] also state that a diverse range of expertise can be gained through self-selecting teams. This is aligned with the findings of our study, which indicated that self-selecting teams involve the right mix of expertise and composition number of junior and senior members in a team. Balancing expertise and levels of experience within teams is vital for successful expertise coordination.

Olsson et al. (2013) claim that self-selecting teams are challenging and stressful at the beginning of projects. This is aligned with our findings, which indicated that it is hard for team members to decide whom they will work with and what tasks fit their expertise. Our findings revealed three activities required in facilitating self-selection, which indirectly tend to support expertise coordination:

- Setting up cross-functional teams
- Selecting the right team
- Balancing teams

Successful expertise coordination relies on the ability of self-selecting teams to form cross-functional teams. Cross-functional teams consist of team members who complement each other with different expertise, experience and perspectives (Juli, 2012). Without diversity of knowledge and skills, there is no point in coordinating expertise in Agile teams. Therefore, management should provide valuable and valid information to set up cross-functional teams during self-selecting teams.

Smaller team formation is essential for cross-functional teams. Melo, Cruzes, Kon, and Conradi (2013) posit that small teams lead to better communication, alignment, and coordination among team members as well. There is a consistency between our findings and Melo et al. (2013), in which minimizing team size tends to support expertise coordination. Table 3 summarizes how small teams play important roles in coordinating expertise.

This paper also points out three factors that need to be considered in selecting the right team: types of work, team members, and knowledge sharing. By comparing these factors, we found that knowledge sharing is a pivotal factor that Agile team members consider when selecting their team.

Therefore, the self-selected teams should consist of a mixture of expertise. Juli (2012) posits that having team members with different backgrounds and expertise gives them opportunities to learn and view things from different perspectives.

Table 3. Minimizing Team Size for Supporting Expertise Coordination in Agile Teams.

SMALL TEAM CHARACTERISTICS	ROLES IN EXPERTISE COORDINATION
Fewer communication lines	Identifying and sharing expertise easily
Tightly bonded relationship	Inducing awareness of team members' expertise
Accessible	Accessing relevant expertise in a timely manner

Our findings, however, contradict Russell and Goodnight's study (Russell & Goodnight, 2009), which claim that self-selecting teams do not tend to have members of divergent expertise, due to a tendency of team members to choose members who work in close proximity. This is opposed to the findings of our study, which revealed that Agile team members put a priority on knowledge sharing and choose team members who possess knowledge and skills that they can gain from and contribute to enhance the team knowledge. Team member factors such as working well together are also considered for successful knowledge sharing, but these are not restricted to members who work in close proximity.

Even though the empowerment of self-selecting teams belongs to team members, we found that facilitation from coaches and management of organization are still needed in forming teams. These findings are supported by Scott and Pollock (2006), who claim that the facilitation

process during team formation is valuable and leads to an efficient process. Guidance from coaches and management assist Agile teams to form the right team with the right balance of expertise.

This paper presented the need to balance teams during team formation. Balancing teams tends to improve team composition with the right roles and number of team members, as well as well-balanced levels of experience within teams. Our findings are consistent with Cohn (2010), who posits that team composition should reach the right balance of expertise, experience, and diversity of team members.

Balancing teams also leads to good team stability. Middleton and Joyce (2012) claim that a stable team with low staff turnover is prevalent in Agile Software Development projects. As depicted in Table 4, our findings align with previous research such as Melo et al. (2013) and Middleton and Joyce's (2012) study on how a stable team supports expertise coordination in Agile teams.

Table 4. Forming Stable Teams for Supporting Expertise Coordination in Agile Teams.

STABLE TEAM CHARACTERISTICS	ROLES IN EXPERTISE COORDINATION
Stable skill set	Identify team members' expertise easily
Tightly bonded relationship	Increase the team members' sense of responsibilities and commitments toward team performance
Minimal staff turnover	Avoid losing critical knowledge and skills when an expert member leaves the team

A stable team also tends to affect customer engagement in Agile Software Development projects. According to Narayanan, Balasubramaniam, and Swaminathan (2011), reallocation of tasks and work disruption occur when there is high team member turnover. This situation tends to affect the understanding of requirements and development of mutual trust between customers and Agile teams. The expertise dependencies between customers and Agile teams are important to ensure the setting of correct expectations of the software product. The perspective of customers' engagement for coordinating expertise, however, is not covered in this paper.

RESEARCH LIMITATIONS AND FUTURE WORK

Although this study generates important findings of Agile self-selecting teams in supporting expertise coordination, we acknowledge several research limitations.

The first research limitation is the applicability of the theory to all forms of Agile project. We recruited participants without restriction as to which Agile methods have been used in Agile projects. The theory, however, is mostly based on research evidence from Agile projects using Scrum and XP practices. Since Scrum and XP are currently the dominant Agile methods, we believe the findings of this study emerged from the various points of view of those Agile methods.

The next research limitation is the applicability of the self-selecting teams, which is not specific to co-located or distributed Agile projects. We have recruited participants without considering co-located or distributed Agile teams. Therefore, this study represents the self-selecting teams in a broad spectrum of Agile teams.

A further research limitation is concerned with the range of research participants. The participants were selected and recruited based on accessibility, thus mostly based in Wellington, New Zealand, and several locations of Agile conferences and workshops. This restricted the range of participants, particularly their working practices, culture, and experiences, which must influence our findings.

In order to mitigate the limitations of the range of research participants, constant comparison and theoretical sampling were used to determine future data required, based on the current data analysis of this study. This is important to ensure the comprehensive nature of data.

A further research limitation is related to data triangulation. We managed to carry out observations at only one company, due to difficulties and challenges in getting Agile Software Development companies to take part in this study. We conducted six observational sessions on two different software development projects over two months, and consequently we had limited observation data that support self-selecting teams. Most of findings on self-selecting teams were emerged from semi-structured interviews.

Future work is essential to confirm the emergent activities of self-selecting teams by using other research approaches, either qualitative or quantitative. These qualitative approaches and other quantitative methods also can be used to test the formulated hypotheses derived from this study either the self-selecting teams activities are strongly influenced the expertise coordination in the context of Agile software development.

There is potential for further Grounded Theory research to explore self-selecting teams in Agile Software Development Projects more depth. Further research could explore more activities or strategies involved in self-selecting teams tend to influence expertise coordiation in Agile Software Development projects.

CONCLUSION

This paper presents three activities involved in facilitating self-selecting teams for Agile software development projects; setting up cross-functional teams, selecting the right team, and balancing teams. Our findings indicated that the activities involved tend to influence the successful expertise coordination in Agile teams. Self-selecting teams are essential to supporting expertise coordination by increasing inter-dependencies between Agile team members, and ensuring a diverse range of knowledge and skills in teams. Through self-selecting teams, we found that several characteristics of teams such as smaller team formation, as well as stable teams, lead to successful expertise coordination.

This study contributes to the body of knowledge by identifying the way of self-selecting teams support expertise coordination. Self-selecting teams are essential to supporting expertise coordination by increasing inter-dependencies between Agile team members, and ensuring a diverse range of knowledge and skills in teams. We revealed how to form self-selecting teams in supporting expertise coordination: setting up cross-functional teams, selecting the right team, and balancing teams. Through self-selecting teams, we found that several characteristics of teams such as smaller team formation, as well as stable teams, lead to successful expertisecoordination. This study contributes to the body of knowledge by identifying the way of self-selecting teams support expertise coordination.

This study also has implications for management in supporting expertise coordination. Management needs to emphasize self-selecting teams, rather than just selecting the available people to join in a particular software project. The self-selecting team activities can be used as a guideline to Agile software organizations in forming self-selecting teams in the fastest and most efficient way. It is vital for management to facilitate the process of self-selecting teams in order to optimize successful expertise coordination. On the other hand, Agile practitioners can consider several factors in choosing the right team: types of work, team members, and knowledge sharing. Agile practitioners are advised to put a priority on knowledge sharing in supporting expertise coordination.

REFERENCES

Abrahamsson, P., Conboy, K., & Wang, X. (2009). 'Lots done, more to do': The current state of agile systems development research. *European Journal of Information Systems, 18*(2009). Palgrave Macmillan Ltd. https://doi.org/10.1057/ejis.2009.27

Allan, G. (2003). A critique of using grounded theory as a research method. *Electronic Journal of Business Research Methods*, *2*(1), 1-10. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.464.1384&rep=rep1&type=pdf

Akgun, A. E., Byrne, J. C., Keskin, H., & Lynn, G. S. (2006). Transactive memory system in new product development teams. *IEEE Transactions on Engineering Management, 53*(1), 95–111. https://doi.org/10.1109/TEM.2005.857570

Akgun, A. E., Byrne, J., Keskin, H., Lynn, G. S., & Imamoglu, S. Z. (2005). Knowledge networks in new product development projects: A transactive memory perspective. *Information & Management, 42*(8), 1105–1120. https://doi.org/10.1016/j.im.2005.01.001

Baker, S. E., Edwards, R., & Doidge, M. (2012). How many qualitative interviews is enough? Expert voices and early career reflections on sampling and cases in qualitative research. *National Centre for Research Methods Review Paper*. Retrieved from http://eprints.brighton.ac.uk/11632/1/how_many_interviews.pdf

Blankenship, M. (2018). How to introduce self-forming teams to your organization. *GitPrime Blog: Perspectives in Engineering*. Retrieved November 10, 2018 from https://blog.gitprime.com/introduce-self-forming-teams-organization/

Birks, M., & Mills, J. (2011). *Grounded theory: A practical guide*. Sage Publications Limited.

Bowen, G. (2009). Document analysis as a qualitative research method. *Qualitative Research Journal, 9*(2), 27–40. Emerald Group Publishing Limited. https://doi.org/10.3316/QRJ0902027

Charmaz, K. (2006). *Constructing grounded theory: A practical guide through qualitative analysis*. Sage Publications Limited.

Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *Computer, 34*(11), 131-133. https://doi.org/10.1109/2.963450

Cohn, M. (2010). *Succeeding with agile: Software development using Scrum*. Pearson Education.

Colomo-Palacios, R., González-Carrasco, I., López-Cuadrado, J. L., & García-Crespo, Á. (2012). ReSySTER: A hybrid recommender system for Scrum team roles based on fuzzy and rough sets. *International Journal of Applied Mathematics and Computer Science*, *22*(4), 801-816. https://doi.org/10.2478/v10006-012-0059-9

Corbin, J. M., & Strauss, A. (1990). Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology,* 13(1), 3-21. https://doi.org/10.1007/BF00988593

Creswell, J. W. (2012). *Qualitative inquiry and research design: Choosing among five approaches*. Sage Publications Limited.

Crowston, K., Annabi, H., Howison, J., & Masango, C. (2004). Effective work practices for software engineering: Free/libre open source software development. In *Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research, ACM,* (pp. 18–26). https://doi.org/10.1145/1029997.1030003

Dorairaj, S., Noble, J., & Allan, G. (2013). Agile software development with distributed teams: Senior management support. In *2013 IEEE 8th International Conference on Global Software Engineering (ICGSE),* (pp. 197–205). https://doi.org/10.1109/ICGSE.2013.33

Faraj, S., & Sproull, L. (2000). Coordinating expertise in software development teams. *Management Science*, *46*(12), 1554-1568. https://doi.org/10.1287/mnsc.46.12.1554.12072

Fowler, M., & Highsmith, J. (2001). The agile manifesto. *Software Development, 9*(8), 28–35. Retrieved from http://users.jyu.fi/~mieijala/kandimateriaali/Agile-Manifesto.pdf

Glaser, B. G. (1978). *Theoretical sensitivity: Advances in the methodology of grounded theory (Volume 2)*. Mill Valley, CA: Sociology Press.

Glaser, B. G. (1992). *Emergence vs forcing: Basics of grounded theory analysis*. Sociology Press.

Glaser, B. G., & Holton, J. (2004, May). Remodelling grounded theory. *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research*, *5*(2), 1-22. Retrieved from www.qualitative-research.net/index.php/fqs/article/download/607/1316

Glaser, B. G., & Strauss, A. L.(1967). *The discovery of grounded theory: Strategies for qualitative research*. Aldine de Gruyter.

Guba, E. G., & Lincoln, Y. S. (1994). Competing paradigms in qualitative research. In N. K. Denzin & Y. S. Lincoln (Eds.), *Handbook of qualitative research* (pp. 105-117). London: Sage. Retrieved from https://www.researchgate.net/profile/Sandra_Richardson2/post/What_does_it_means_to_strengthen_theoretical_links/attachment/59d6213e79197b807797fa52/AS:295415858647041@1447444037342/download/10-guba_lincoln_94.pdf

Hoda, R., Noble, J., & Marshall, S. (2012). Developing a grounded theory to explain the practices of self-organizing agile teams. *Empirical Software Engineering, 17*(6), 609–639. https://doi.org/10.1007/s10664-011-9161-0

Juli, T. (2012). *The power and illusion of self-organizing teams*. Project Management Institute.

Kirk, J., & Miller. M. L. (1986). *Reliability and validity in qualitative research*. Sage Publications Limited. https://doi.org/10.4135/9781412985659

Kolb, S. M. (2012). Grounded theory and the constant comparative method: Valid research strategies for educators. *Journal of Emerging Trends in Educational Research and Policy Studies, 3*(1), 83–86. Retrieved from http://jeteraps.scholarlinkresearch.com/articles/Grounded%20Theory%20and%20the%20Constant%20Comparative%20Method.pdf

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments: A brief history. *Computer, 36*(6), 47–56. https://doi.org/10.1109/MC.2003.1204375

Levesque, L. L., Wilson, J. M., & Wholey, D. R. (2001). Cognitive divergence and shared mental models in software development project teams. *Journal of Organizational Behaviour, 22*(2), 135–144. https://doi.org/10.1002/job.87

Lewis, K. (2003). Measuring transactive memory systems in the field: scale development and validation. *Journal of Applied Psychology 88*(4), 587. https://doi.org/10.1037/0021-9010.88.4.587

Magnes, D. (2016, July 11). Self-selecting on small-scale. *Medium*. Retrieved from https://medium.com/@MagnusDahlgren/self-selecting-teams-on-a-small-scale-b5434089e102

Melo, C. D. O., Cruzes, D. S., Kon, F., & Conradi, R. (2013). Interpretative case studies on agile team productivity and management. *Information and Software Technology*, *55*(2), 412–427. https://doi.org/10.1016/j.infsof.2012.09.004

Middleton, P., & Joyce, D. (2012). Lean software management: BBC worldwide case study. *IEEE Transactions on Engineering Management, 59*(1), 20–32. https://doi.org/10.1109/TEM.2010.2081675

Narayanan, S., Balasubramaniam, S., & Swaminathan, J. M. (2011). Managing outsourced software projects: An analysis of project performance and customer satisfaction. *Production and Operations Management, 20*(4), 508–521. https://doi.org/10.1111/j.1937-5956.2010.01162.x

Olsson, H. H., Bosch, J., & Alahyari, H. (2013). Towards R&D as innovation experiment systems: A framework for moving beyond agile software development. In *Proceedings of the IASTED*, (pp. 798–805). https://doi.org/10.2316/P.2013.796-008

Parry, K. W. (1998). Grounded theory and social process: A new direction for leadership research. *The Leadership Quarterly 9*(1), 85–105. https://doi.org/10.1016/S1048-9843(98)90043-1

Rejab, M., Noble, J., & Allan, G. (2014a). Distributing expertise in agile software development projects. In *2014 IEEE Agile Conference (AGILE)* (pp. 33–36). https://doi.org/10.1109/AGILE.2014.16

Rejab, M., Noble, J., & Allan, G. (2014b). Locating expertise in agile software development projects. In G. Cantone, & M. Marchesi (Eds.), *Agile Processes in Software Engineering and Extreme Programming,* (pp. 60–268). Springer. https://doi.org/10.1007/978-3-319-06862-6_19

Rejab, M. M., Noble, J., & Marshall, S. (2015). Coordinating expertise outside agile teams. In *International Conference on Agile Software Development*, (pp. 141-153). Springer, Cham.

Rico, R., Sánchez-Manzanares, M., Gil, F., & Gibson, C. (2008). Team implicit coordination processes: A team knowledge-based approach. *Academy of Management Review, 33*(1), 163–184. https://doi.org/10.5465/amr.2008.27751276

Russell, D. L., & Goodnight, J. E. (2009). The rolling learning cell: A method to integrate individual assessment and team grading components in information systems curriculum team projects. *Information Systems Education Journal* , 7(37), 1–15.

Sandy, M., & David, M. (2014). Self-selecting teams part 1 - Why you should try self-selection. *Software Development Magazine – Project Management, Programming, Software Testing*. Retrieved January 10, 2018 from http://www.methodsandtools.com/archive/selfselectingteams.php

Scott, E., & Pollock, M. (2006). Effectiveness of self-selected teams: A systems development project experience. *Issues in Informing Science and Information Technology, 3*, 601–617. https://doi.org/10.28945/918

Stern, P. N., & Porr, C. (2010). *Essentials of accessible grounded theory*. Left Coast Press.

Strauss, A., & Corbin, J. (1998). *Basics of qualitative research: Procedures and techniques for developing grounded theory*. Thousand Oaks, CA: Sage Publications Limited.

Strode, D. E., & Huff, S. L. (2012). A taxonomy of dependencies in agile software development. In *ACIS 2012: Proceedings of the 23rd Australasian Conference on Information Systems, ACIS*, (pp. 1–10).

Tuckman, B. (1965). Developmental sequence in small groups. *Psychological Bulletin 63*(6), 384–99. https://doi.org/10.1037/h0022100

Urquhart, C., Lehmann, H., & Myers, M. D. (2009). Putting the 'theory' back into grounded theory: Guidelines for grounded theory studies in information systems. *Information Systems Journal 20*(4), 357–381. https://doi.org/10.1111/j.1365-2575.2009.00328.x

Wegner, D. M. (1987). Transactive memory: A contemporary analysis of the croup mind. In B. Mullen, & G. R. Goethals (Eds.) *Theories of group behaviour* (pp. 185-208). Springer Series in Social Psychology. New York, NY: Springer. https://doi.org/10.1007/978-1-4612-4634-3_9

Wegner, D. M. (1995). A computer network model of human transactive memory. *Social Cognition, 13*(3), 319–339. https://doi.org/10.1521/soco.1995.13.3.319

Waterman, M., Noble, J., & Allan, G. (2015). How much up-front? A grounded theory of agile architecture, In *2015 IEEE/ACM 37th IEEE International Conference Software Engineering (ICSE), 1*, 347–357.

Weber, R. (2012). Evaluating and developing theories in the information systems discipline. *Journal of the Association for Information Systems 13*(1), 1–30. https://doi.org/10.17705/1jais.00284

BIOGRAPHIES

Mawarny Md Rejab is senior lecturer of School Computing at Universiti Utara Malaysia. She completed her doctoral degrees at Victoria University of Wellington. Her PhD thesis was in the field of software engineering specializing in Agile software development. She has published papers and is involved in research on software process, software testing, problem-based learning, and grounded theory.

James Noble is Professor of Computer Science and Software Engineering at Victoria University New Zealand. He has published many papers on object-orientation design and programming, aspects, software visualisation and software engineering in international academic conferences and journals. He is the author of Small Memory Systems: Patterns for Systems with Limited Memory (with Charles Weir), editor of Prototype-Based Programming, Pattern Languages of Program Design 5, and Aliasing in Object-Oriented Programming (with various co-editors), a Foundation Editor-In-Chief of Springer-Verlag Transactions on Pattern Languages of Progrmaming, and on the Editorial Boards of IET Software, Systems Signs and Actions, and the International Journal of Agile and Extreme Software Development. He has been on the Programme Committees of a number of conferences such as ECOOP (including program chair) OOPSLA (including Chair of Onward! and the Dynamic Languages and Doctoral Symposia), AOSD, TOOLS, ACCPM, AUIC, ACSC, CATS, EuroPLOP, KoalaPLoP (including as chair), and VL/HCC.

Stuart Marshall is the Head of School for Engineering and Computer Science at Victoria University of Wellington. Stuart is also a Senior Lecturer in the School and his research interests are in information visualisation, m-learning and agile software methodologies. Stuart has published in numerous international journals and conferences and is the Vice-Chair of the New Zealand Chapter of the Association of Computer Machinery (ACM) – Special Interest Group on Computer Human Interaction (SIGCHI). Stuart has also been on the programme committees for a variety of international conferences in human-computer interaction, software engineering requirements analysis, as well as more locally in the Australasian Software Engineering Conference series.