Agility framework for software development: an investigation into agility concepts in the software development industry

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF COMPUTER AND INFORMATION SCIENCES

Supervisor:

Dr Ramesh Lal

2018

Kevin Kusuma

Auckland University of Technology

School of Engineering, Computer and Mathematical Sciences

Abstract

Context: With agile software development, agile values and principles (stated in the agile manifesto) may lead to agility with software development. In this study, agile organization concepts are investigated through the survey method with agile software development to identify agile practices that will support agile values and principles to guide adoption and adaptation of agile processes and method practices for achieving agility.

Objective: This thesis shows agile organization concepts are influencing agile software development through practices to support agile values and principles to help shape the agile development environment for teams to achieve software development agility. Through this study, eight agile organization concepts (knowledge management, organizational culture, organizational learning, competencies, responsiveness, speed, team effort and workforce agility) and their relating agile practices are identified for driving the agile software development environment for agility.

Method: A quantitative approach involving the survey method was used for this investigation. The survey questions were developed and tested based on the literature review on agile organizations and agile software development. A list of possible participants, consisting of agile software development practitioners (software vendors, in-house software development teams of business organizations or institutions and software development contracting companies) was compiled, and they were invited to take part in this investigation. The statistical analysis tool, SmartPLS was used to conduct the analysis of the data collected.

Results: The research reveals eight (8) agile organization concepts that are critical factors driving agile software development for achieving agility. The results of the survey identify organizational learning as the most critical agile organization concept for agility. Hence, learning relating to product development and management in a software development environment is as critical as producing the actual software. Through the survey results, based on the eight agile organization concepts, a number of agile software development practices have been identified which can guide adoption and adaptation of agile method practices for gaining agility with software development.

Conclusion: This study investigated the agile organization concepts that influence agility capability in the software development environment. From this investigation, it can be concluded that agile organization concepts must be recognized as critical for shaping agile values and principles, so that achieving agility with software development is reinforced as the mind-set for agile method practitioners.

Acknowledgments

First, I would like to express my deepest appreciation to my supervisor, Dr Ramesh Lal for the continuous support for me during the completion of this thesis; especially for his patience, enthusiasm, motivation and immense knowledge. During the hard times, he steered me in the right direction and always provided guidance whenever I needed it. I could not have imagined having a better advisor and mentor for my thesis.

I would also like to thank the participants who have willingly shared their precious time in the validation survey. The validation survey could not have been successfully conducted without their passionate participation and input. Individual and company details cannot be revealed to protect their anonymity.

A sincere thank you for Gary Ferguson for proofreading this thesis. He helped me to proofread this thesis in two weeks and he was professional, extremely organized and reliable. I would recommend Gary to anybody seeking proofreading service in the future.

Finally, I must express my very profound gratitude to my parents, sisters and friends for providing me with unfailing support and continuous encouragement through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

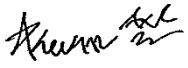
Table of Contents

List of Tables

List of Figures

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Signed:  Date: 8 December 2017

Chapter One

1.0 Introduction

Agile approaches have become the mainstream software development method (West & Grant, 2010); hence, adopting it ought to result in software development agility (Danesh, 2011). Agility capability enables software development organizations and development teams to have the ability to deal with unpredictable situations and rapid changes in their business environment faster than their competitors (Highsmith, 2002).

Most of the studies undertaken to investigate an agile approach for software development focus on method adoption. To the best of knowledge, there are just two that investigate agility. Lee and Xia (2010) investigated agility concepts, but their study was limited to team autonomy, team diversity and responsiveness ability. However, there are several other agility concepts which are important characteristics of agile organizations (Lee & Xia, 2010; Sherehiy, Karwowski, & Layer, 2007). Hence, this study investigates a range of agility concepts based on agile organization characteristics to identify and provide understanding of agile software development practices that enable agility. Knowing and understanding these agility concepts together with agile values and principles are critical (Highsmith, 2002), as they provide development practices enabling software development organizations and development teams to have a strategic software development process in a market-driven environment.

1.1 Agility

The concept of agility comes from the organizational psychology discipline (Goldman & Nagel, 1993). Agility is defined as the ability of organizations to respond to business challenges and to deal with rapidly changing global markets to provide high-quality and high-performing products or services, including proving customer-configured products and services (Bernardes & Hanna, 2009).

1.2 Agility Benefits

Organizational agility offers several benefits. Agility capability enables identification and securing of market opportunities with speed and competitive action (Sambamurthy, Bharadwaj, & Grover, 2003). In addition, agility provides the capability for organizations to use market knowledge and create virtual corporations to enhance opportunities in a volatile market place (Mason-Jones, Naylor, & Towill, 2000). Furthermore, agility acts

as an overall strategy for organizations to respond efficiently in uncertain environments (Sanchez & Nagi, 2001).

Moreover, agile organizations adapt to meet customer demand and rapidly respond to changes in an unpredictable market (Christopher, 2000). Agility capability not only enables taking advantage of changes, but also responding to change in a competitive way to deal with uncertainty (Sharifi & Zhang, 1999; Sherehiy et al., 2007). Importantly, agility enables more innovative responses to unpredictable changes (Wadhwa & Rao, 2003).

Finally, agile organizations have the ability to become competitive by enriching their customers, through cooperation, mastering changes and leveraging the impact of people and information (Goldman, Nagel, & Preiss, 1995).

Thus, agility is critical for software development organizations and teams to have an on-going capability to learn and adapt their processes, practices, roles and skills in their software development environment.

1.3 Structured and Agile Organizations

Mechanistic or structured organizations develop standardized products, emphasize development efficiency and compete based on product cost, quality and delivery (House, 1991). These organizations are hierarchical, structured and procedural while being driven by order and control (Wallach, 1983). Due to their development efficiency, these organizations are regarded as machine-like organizations and individual behaviour is predetermined (Burns & Stalker, 1994). However, such organizational behaviour is counter-productive in a market-driven environment.

In contrast, organic (agile) organizations operate in an unpredictable and changing environment, adapt to the environmental demands and are innovative and flexible (House, 1991). According to House (1991), they compete based on new features and products, product customization and process and technology innovations. These are constantly changing and adapting organizations (Burns & Stalker, 1994). Such organizational behaviour is required to be able to deliver innovative products. The organizational culture is an outcome of its mechanistic or organic (agile) structure (Burns & Stalker, 1994). Therefore, there are certain agile concepts that ought to be adopted that lead to an agile culture and eventually, to agility.

1.4 Agile Organization Concepts

Table 1 provides a list of agility concepts that define agile organizations. These concepts have been compiled from agile organization literature. These agile organization concepts enable flexibility and continuous learning to have the ability to react to changes and meet the market requirements

Table 1. Agile Organization Concepts and Their Definitions

Agile organization concepts	Definition
1. Workforce agility	Learning and self-development, problem-solving ability and ability to generate innovative ideas (Plonka, 1997).
2. Teamwork skilled workforce	Knowledge in teamwork and work in multi-functional workforce (Gunasekaran, 1999).
3. Responsiveness attributes	• Flexibility to respond to changes, quick to adapt if change happens and swiftly upskill (Breu, Hemingway, Strathern, & Bridger, 2002). • The ability to identify changes and respond quickly (Sharifi & Zhang, 1999).
4. Interactive and communication skills	Ability for spontaneous collaboration and work in multiple roles (Dyer & Shafer, 2003).
5. Information management ability with inter-organization cooperation	Utilize the network connection and have strategic partnership to create high quality information (Lin, Chiu, & Chu, 2006).
6. Competency	Wider abilities to improve productivity (efficiency and effectiveness) (Sharifi & Zhang, 1999)
7. Flexibility	The ability to change direction and achieve different objectives (Sharifi & Zhang, 1999)
8. Speed	The ability to perform tasks in shortest possible time (Sharifi & Zhang, 1999).
9. Team building	Empowered individuals that can work in cross-functional teams and are able to make effective decisions (Yusuf, Sarhadi, & Gunasekaran, 1999).
10. Organizational culture	Supportive environment that enables employees to perform continuous improvement and innovation and have the ability to re-configure (Sherehiy et al., 2007).

For this study, the agility concepts investigated include: (1) Knowledge Management; (2) Organizational Culture; (3) Organizational Learning; (4) Competencies; (5) Responsiveness; (6) Speed; (7) Team Effort (Teamwork skilled workforce and Team building); (8) Workforce Agility.

1.5 The Importance of Agility in Software Development

One critical reason for software development agility is the challenge now to swiftly learn and capture the specific requirements to be implemented. This is due to the nature of the requirements, impacted by the current business environment and emerging technologies (Schmidt, Lyytinen, & Mark Keil, 2001). Hence, software development teams must have the ability to identify, learn, improve and use different tools, techniques and processes on the fly to able to identify, verify and validate business value requirements. According to Clancy (2014), incomplete requirements and lack of user involvement and resources are the main reasons why most projects are cancelled.

In a survey report conducted in 2009 by the Economist Intelligence Unit shows that 40% of respondents considered agility as an "*extremely important*" factor contributing to the overall success of an organization. This shows that there is a growing awareness amongst businesses of agility capability in driving business success. Therefore, the key lies in the ability of software development teams to anticipate and adapt to change to enable their business organization to excel in a volatile environment through IT and software products.

A report produced in 2012 by the Project Management Institute shows that 75% of participants described "the ability to rapidly respond to strategic opportunities" as the main organizational agility characteristic, while 64% of respondents stated the ability to have shorter production, reviews or decision life cycles as the main characteristic of organizational agility. These capabilities are only possible through having agility in software development.

As a result, organizations have to understand and manage an emerging concept like agility that could bring benefits for their business such as implementing best practices in terms of change and risk management and also to make change standard for business programmes, portfolios and project management practices (Project Management Institute, 2012).

Researchers have investigated identifying agility characteristics of organizations (Christopher, 2000; Gunasekaran, 1998; Sherehiy et al., 2007; Yusuf et al., 1999). However, there is still little information and understanding on whether adoption of the agile software development approach has led to agility within the adopters in their software development environment, including whether they have realised the perceived agile benefits.

1.6 Background

Agile Approach for Software Development

The agile approach for software development enables development teams to effectively deliver projects through short development cycles allowing planning and development through prioritization (Lin, Chiu, & Tseng, 2006). In addition, this approach allows learning and dealing with risks continuously throughout the project with an iterative approach rather than changing the model all at once and enabling the development team to build the product from the most important requirements (Sedehi & Martano, 2012).

The agile approach for software development incorporates both managerial and technical practices with the emphasis on dealing with changes and risks throughout a project's timeline.(Turk, France, & Rumpe, 2014). The agile philosophy of short development cycles allows development teams to deliver value for the customer through frequent and regular delivery throughout the project (Mohammadi, Nikkhahan, & Sohrabi, 2008). In addition, short development cycles provide the opportunity for reviews, including providing the ability to make changes to the implementation decisions, ensuring stakeholder satisfaction (Cockburn & Highsmith, 2001). The agile idea for a business role (implemented through an on-site customer role) and development team to work closely on a daily basis provides development capability to deliver features according to the exact needs of the business (Koskela & Abrahamsson, 2004).

Agility in Software Development

According to Henderson-Seller & Serour (2005), agility with software development means having the capability to respond to and deal with changes in the product development environment, which includes having the ability to adapt processes, practices and roles when changes happen during the project timeline. In addition, software development agility enables the ability to swiftly learn from the change and improve value for customers through simplicity and quality (Drury, Conboy, & Power, 2012).

The motivation for agility in software development is same as the motivation for organizations to be agile organizations. Software development needs to adopt the agility philosophy in order to survive in the current market-driven environment characterised by emerging technologies and change in client/end-user behaviour.

Four Agile Values and Twelve Agile Principles

The agile approach for software development has the Agile Manifesto (2001) that identifies the four values and twelve principles for agile software development (Schmidt, 2015). According to Lal (2011), the agile values symbolise its fundamental viewpoints for successful software development. These teams ought to have continuous interaction amongst key individuals when designing and developing software, including having an on-going collaboration with the stakeholder (client) for feedback (Selleri Silva et al., 2015). In addition, development teams must deliver client value through providing a working code on a continuous basis and respond to their requests for change throughout the project. The twelve principles provide strategies to achieve the agile values to ensure the successful development (Lal, 2011). These agile principles ought to guide small or large-scale software developments (Antanovich, Sheyko, & Katumba, 2010; Eklund & Bosch, 2012; Katumba & Knauss, 2014).

Table 2. 4 Agile Values and 12 Principles

(Adopted from https://www.agilealliance.org/agile101/the-agile-manifesto/)

Agile Values:
1. Individuals and Interactions over Processes and Tools 2. Working Software over Comprehensive Documentation 3. Customer Collaboration over Contract Negotiation 4. Responding to Change over Following a Plan
Agile Principles:
1. Customer satisfaction through early and continuous software delivery 2. Accommodate changing requirements throughout the development process 3. Frequent delivery of working software 4. Collaboration between the business stakeholders and developers throughout the project 5. Support, trust, and motivate the people involved 6. Enable face-to-face interactions 7. Working software is the primary measure of progress 8. Agile processes to support a consistent development pace 9. Attention to technical detail and design enhances agility 10. Simplicity 11. Self-organizing teams encourage great architectures, requirements, and designs 12. Regular reflections on how to become more effective

Agile Methods

Several software development methods come under the agile banner. These methods are driven by four values and twelve principles. However, these methods also have their own phases and practices. Table 3 identifies most of widely used agile methods.

Table 3. List of Single Agile Methods

Single Agile Method	Creator/Author	Lifecycle/Phases
1. Extreme Programming (XP)	Kent Beck	• Planning • Design • Coding • Testing
2. Scrum	Ken Schwaber Jeff Sutherland	• Pre-game (Planning) • Development (Sprint cycles) • Post-game (Release)
3. Adaptive Software Development (ASD)	Jim Highsmith Sam Bayer	• Speculate (Initiation and Planning) • Collaborate (Concurrent feature development) • Learn (Quality review)
4. Rapid Application Development (RAD)	James Martin	• Requirements planning phase • User design phase • Construction phase • Cutover phase
5. Feature Driven Development (FDD)	Jeff Luca	• Develop an Overall Model • Build a Features List • Plan by Feature • Design by Feature • Build by Feature
6. Crystal Methods	Alistair Cockburn	No clear processes defined, however the methodology can be divided into five different methods to develop a strategy with different team sizes, system criticalities, and project priorities.
7. Dynamic Systems Development Method (DSDM)	Association of vendors and experts in software engineering from British Airways, American Express, Oracle, and Logica	• Pre-project • Feasibility • Foundations • Evolutionary development • Deployment • Post-project
8. Kanban	Inspired by Toyota Production System	• Backlog • Requirements • Design • Testing • Development • Deployment • User Acceptance Testing • Done

The Extreme Programming (XP) and Scrum methods appear to be the two most adopted agile methods (VersionOne, 2016).

The reasons why the Scrum method is popular are as follows:

1. Because of its simplicity, the Scrum method enables the development teams to gain direction through implementation and allows them to improve the response to changes (Maximini, 2015).

2. The Scrum method provides effective and efficient planning practices to plan and manage projects including the ability to deal with emerging risks and issues, and change during the project to deliver the project successfully (Guang-yong, 2011).
3. Development teams learn continuously from the previous sprints through retrospective meetings and team members are encouraged to actively participate in the meetings and tasks so that the project can be delivered successfully (Sandberg & Crnkovic, 2017).

The XP method is another most adopted agile method due to the following reasons:

1. XP has useful agile practices such as iterative development, velocity tracking and user stories, which create visibility and easily monitor progress on a daily basis (Srinivasan & Lundqvist, 2009).
2. XP employs an on-site customer role that encourages developers to work closely and have effective collaboration on the critical tasks in projects with them so that right decisions can quickly be made (Zhou, 2009).
3. With the XP method, a project can be divided and undertaken by sub-teams enabling development in parallel. (Zeng, Wang, & Long, 2009).

Hybrid Agile Method

Instead of adopting the entire agile method, development teams can also create a hybrid agile method based on selecting relevant practices from different agile methods, including prices from a structured method. However, the adoption of selected practices must be driven by the agile manifesto. Hence, a hybrid agile method is made of practices selected from at least one agile method together with practices from other agile or structured methods.

Table 4 below lists some of the hybrid agile methods based on investigations done by different researchers on agile software development. It appears that the XP and Scrum methods practices are common to create a hybrid agile method.

Table 4. Example of Hybrid Method and The Agile Practices

Hybrid Agile Methods	Author	Agile Practices
1. XP Scrum DSDM	(Sultana, Motla, Asghar, Jamal, & Azad, 2014)	• Coding Standard • Test Driven Development • Pair Programming • Continuous Integration • Product Backlog • Retrospectives • Acceptance Testing

		• Unit Testing • Code Refactoring • Sprint Planning
2. XP **FDD**	(Doshi & Patil, 2016)	• Pair Programming • Test Driven Development • Continuous Integration • Coding Standard • Release Planning • Iteration Planning (Sprint Planning) • Collective Code Ownership
3. Scrum **UCD**	(Anwar et al., 2014)	• Retrospective • Release Planning • Sprint Planning • Product Backlog • Daily Stand-up Meeting • Coding Standard • Unit Testing • Continuous Integration
4. Scrum **XP** **Kanban**	(Bougroun, Zeaaraoui, & Bouchentouf, 2014)	• Product Backlog • Pair Programming • Sprint Planning • Continuous Integration • Retrospective • Collective Code Ownership • Open Workspace
5. Scrum **XP**	(Braz, Rubira, & Vieira, 2015)	• Release Planning • Sprint Planning • Product Backlog • Retrospective • Continuous Integration • Acceptance Testing • Daily Stand-up Meeting • Pair Programming • User Story
6. XP **Scrum** **Crystal**	(Shi, Chen, & Chen, 2011)	• Pair Programming • Test Driven Development • Retrospective • Continuous Integration • Release Planning • Sprint Planning • Product Backlog • Daily Stand-up Meeting

Scaled Agile Method

Agile methods were originally considered as the best suited for small software development projects with co-located teams (Abrahamsson, Conboy, & Wang, 2009; Williams & Cockburn, 2003). However, there is a dilemma in terms of scaling these methods to be used in large projects where multiple development teams work together.

Large development teams use the scaled agile method to build software functionalities quickly without sacrificing the underlying values and principles stated in the Agile Manifesto (Reifer, Maurer, & Erdogmus, 2003). The main challenge of deploying the scaled agile method in organizations is that the agile method must be applied where both traditional and agile methods can be used together (Reifer et al., 2003).

Table 5. Scaled Agile Methods (Alqudah & Razali, 2016)

Scaled Agile Methods	Creator/Author	Lifecycle/Phases
1. Disciplined Agile Delivery (DAD)	Scott Ambler Mark Lines	• Roadmap • Scope • Design • Implement • Integrate • Release • Support
2. Scaled Agile Framework (SAFe)	Dean Leffingwell	• Take an economic view • Actively manage queues • Understand and exploit variability • Reduce batch sizes • Apply work in process (WIP) constraints • Control flow under uncertainty through cadence and synchronization • Get feedback as fast as possible • Decentralize control
3. Large Scale Scrum (LeSS)	Bas Vodde Craig Larman	• Sprint Planning Part 1 • Sprint Planning Part 2 • Daily Scrum • Coordination • Overall PBR • Product Backlog Refinement • Sprint Review • Overall Retrospective
4. Nexus	Ken Schwaber	• Refine the Product Backlog • Nexus Sprint Planning • Development Work • Nexus Daily Scrum • Nexus Sprint Review • Nexus Sprint Retrospective
5. Recipe for Agile Governance (RAGE)	Kevin Thompson	• Business • Governance • Portfolio • Program • Project • Delivery

Discipline Agile Delivery (DAD) and Scaled Agile Framework (SAFe) are the two most popular scaled agile methods (Laanti, 2014). DAD is a process decision framework that is goal-driven for phase and also uses the hybrid agile method where

practices are adopted from existing agile method such as Kanban, Extreme Programming, Scrum, and agile modelling etc. (Brown, Ambler, & Royce, 2013). The DAD framework provides guidance for the development team to develop a strategy that reflects the current situation and chooses the best approach to successfully apply agile in practice (Ambler & Lines, 2012); while SAFe aims to provide a continuous flow of value to the customer, end-user or business based on the development and deployment process to create a long-lived system (Vaidya, 2014). In addition, SAFe presents guidelines on how to plan the releases, while the individual teams work and deliver iterations (Eklund, Olsson, & Strøm, 2014).

1.7 Research Objectives

The main objectives of this research study were to learn agility practices directly from the agile software development teams and organizations to provide a list of agile method practices that lead to agility and to provide an understanding of agility elements in software development.

The research question for this study is as follows:

"What are the software development agility practices based on agile organization concepts?"

In order to ensure that this research question is relevant and valid, leading to a desired outcome the following steps were undertaken:

1. Key constructs (agility practices) for the study were identified through investigation of the literature in regard to agile organizations and agile software development method practices.
2. Identify seven hypotheses based on the agile software development practices which ought to lead to organizational agility (agile organization concepts).
3. Construct and improve survey questions based on the feedback.
4. Improve the survey questions through a small pilot test.
5. Conduct the survey by identifying and sending emails to the potential participants. Survey questions were hosted on Google Survey.

1.8 Research Method

This study used a quantitative approach to learn agile method practices leading to organizational agility. Hence, a survey technique was used so that responses from a large number of respondents from the agile software development community in New Zealand,

and Australia could be targeted. Respondents were identified through agile community groups and various agile software development conferences held in New Zealand and Australia.

This research used structural equation modelling techniques to analyse the data (Hair, Ringle, & Sarstedt, 2011). Smart PLS was used to assess the hypothesis relationship in the structural model, including for reliability and validity of the measurement model. Hence, some of the data analysis techniques used to assess the reliability and validity were as follows: Internal consistency reliability (ICR), Average variance extracted (AVE) for convergent validity, discriminant validity (square root of AVE), coefficient determination (R^2 Values), path coefficients and t-values.

1.9 Outline of the Thesis

The thesis is organized as follows:

Chapter One provided the background, motivation and significance of the study. It included the research objectives and research questions for this study.

Chapter Two discusses the relevant literature of the agile organization and agile practices in software development. It also discusses the main characteristics of an agile organization, and how it differs from an organization that applies a centralized approach.

Chapter Three discusses how the characteristics from an agile organization can provide insight to the agile software development. A set of research hypotheses are proposed together with the research model.

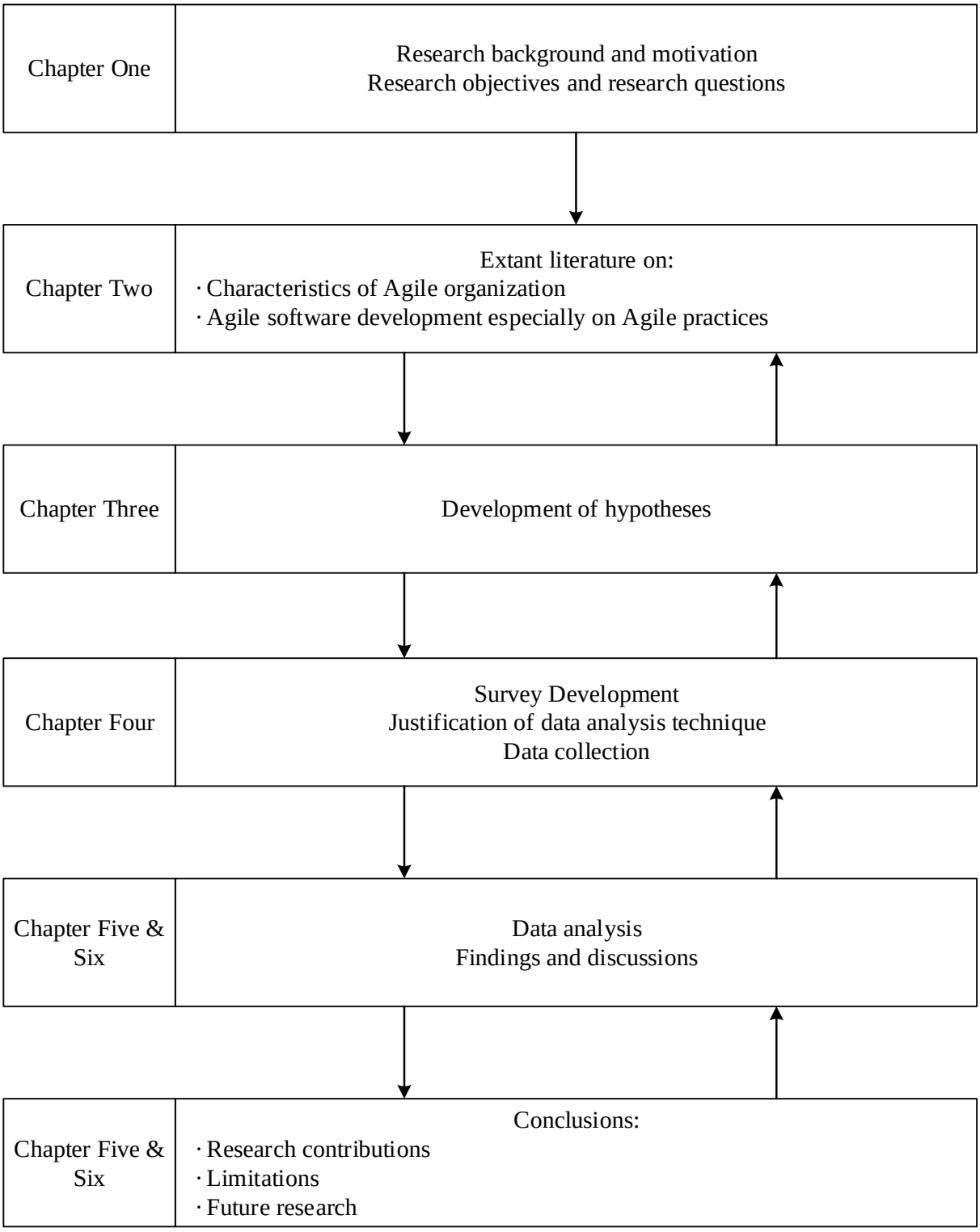
Chapter Four describes the methodology selected for this study. This section elaborates the specification for choosing data collection methods and data analysis techniques as well the reliability and validity of the data.

Chapter Five reports the data analysis procedures and results. This section provides the information from general result of the survey and hypothesis testing results.

Chapter Six discusses the main findings from the data analysis results.

Chapter Seven summarizes the main findings of the study.

1.10 Study Roadmap



Chapter Two: Literature Review

2.0 The Structures of an Agile Organization

The concept of the adaptive organization originally came from the theory of the contingency approach (Miller, 1981). The contingency approach states that there is no single way to manage or organize a company and each organization is unique, having its own approach to solving the problem related to constraint at the organization's operational level (McKenna, 2000).The main objective of the contingency approach is to have understanding in order to achieve better performance results. Organizations should fit themselves between the contingency and structure of the organization (Donaldson, 2001). Thus, organizational strategies, organizational size and the environment play important roles for the organization to maintain the effectiveness and efficiency of its operations.

Following the idea of the contingency approach, Burns and Stalker (1961) identified that there are two main types of organizational structure: mechanistic and organic organizational structures.

The main features of mechanistic organizational structures are organizations that tend to have stable and predictable mechanisms that usually involve standard procedures to assist employees when solving the case of routine operations. This type of organization requires an employee to specialize in one task only and the communication is hierarchical. Hence, employees with lower authority must follow the instructions and be coordinated by upper management. Mechanistic organizations have many rules, standards and operating procedures, and as a result, mechanistic organizations are required to document all business operations (Burns & Stalker, 1994; Donaldson & Siegel, 2001; Pierce & Delbecq, 1977; Zammuto & O'Connor, 1992).

Dynamic organizations are characterized by new challenges and face an unstable environment. This type of organization is also known as an organic organization. This type of organization has employees with multi-functional capabilities while being ready to adapt to change. The communication with and between employees tends to be less formal compared to mechanistic organizations (Burns & Stalker, 1994; Miles, Snow, Meyer, & Coleman, 1978; Nonaka, 1994; Sherehiy et al., 2007).

Goldman et al. (1995) developed a strategy to achieve competitive capabilities leading to organizational agility. Their strategy dimension for gaining agility involves:

(1) enriching the customer; (2) cooperating to enhance competitiveness; (3) organizing to learn changes and (4) leveraging the impact of people and information. The first strategic agility dimension allows organizations to deliver value that could help their customers to solve their business problems rather than just focusing on delivering products. Cooperation is required between organizations in order to deliver products swiftly to the marketplace. Furthermore, organizations are required to have flexible structures to master changes. According to Goldman et al. (1995), agile organizations are able to create the distinction between themselves when compared to their competitors at the marketplace through their product development experience, people and skills. Thus, it is essential to have continuous workforce training and education through formal and informal approaches.

2.1 Workforce Agility

Workforce agility refers to the ability to adapt skills and practices in a short time frame to accept and act on the change, enabling organizational agility (Sohrabi, Asari, & Hozoori, 2014). Hence, the critical reason to have workforce agility is to have the organizational ability to respond appropriately and deal with unpredictable changes that frequently happen at the marketplace (Qin & Nembhard, 2015). According to Sherehiy et al. (2007), workforce agility ought to result in organization benefits such as on product quality, customer service, learning and ability to identify the product scope.

The workforce agility strategy is often ignored in structured organizations because top-level management often themselves coordinate tasks and solve any problems arising from the tasks carried out by employees. Hence, such employees have limited authority, resulting in insufficient knowledge of processes to have problem solving abilities (Tata & Prasad, 2004).

From the agile organization's perspective, workforce agility means having the ability to swiftly solve problems and being able to respond promptly to uncertainties in their business environment. Agile organizations develop and maintain workforces that are skilled and flexible to deal with emerging and non-routine tasks (Youndt, Snell, Dean, & Lepak, 1996). Moreover, Sumukadas and Sawhney (2004) explained that workforce agility is achieved through job rotation, enrichment and teamwork.

Workforce requirements are changing all the time and there is always a need to anticipate future skill requirements for employees (Weick, 1969). Specialization of the

work force prevents organizations from adapting and reacting to market situations swiftly (Dyer & Shafer, 2003; Qin & Nembhard, 2015).

The five attributes of workforce agility are responsiveness, quickness, competence, adaptability and cooperativeness (Qin & Nembhard, 2015). Responsiveness relates to the attitude of workers towards unexpected changes. Quickness capability relates to the swift ability to adapt to new working conditions without any impact on productivity. Competence ability relates to the creativity and ability of workers to solve problems. Adaptability relates to the accepting and performing in required work conditions. Cooperativeness means having collaboration and helping each other when pursuing the common goal when working in different roles and functional units.

Breu et al. (2002) explains that information technology (IT infrastructure and mobile technologies) plays an important role in enabling workforce agility by providing physical, structural and cultural resources to enable responsiveness, speed and flexibility abilities including empowerment in decision making. Breu et al. (2002) also identified intelligence, competencies, collaboration, culture and information systems as the main attributes of workforce agility.

Plonka (1997) investigated the human factors in workforce agility in order for organizations to develop an effective production system for the manufacturing environment, targeting to enhance the capabilities and skills of the roles and individuals. The human factors involved were attitude toward self-development and continuous learning, ability to solve problems, being comfortable with new changes, ideas and technologies, the ability to generate innovative ideas, and accepting new responsibilities (Plonka, 1997).

Griffin & Hesketh (2003) suggest that employees must adapt to have proactive, reactive and tolerant behaviours. According to their study, people with proactive behaviour bring positive impacts to the dynamic environment. Reactive behaviour allows individuals to adapt while tolerant behaviour allows individuals to maintain productivity despite change.

According to Gunasekaran (1999), the important characteristics of an agile workforce are empowerment, multi-functional teams, negotiating ability, knowledge management, self-management teams and IT staff. Sumukadas and Sawhney (2004) developed a theoretical framework concentrating on various types of management practices for employees related to workforce agility such as information sharing, training,

reward and power sharing. Based on the empirical investigation, power sharing was described as the most important attribute enhancing workforce agility through enabling employees to solve problems and provide suggestions (Sumukadas & Sawhney, 2004).

According to Hopp and Van Oyen (2004), workforce agility enables organizations to deliver products or services providing customer value based on cost, time, quality and variety. Hence, workforce agility enables organizations to have higher productivity and responsive employees, quality improvement and capability to provide more products or services.

Dyer and Shafer (2003) identified proactive, adaptive and generative behaviours as key to the agility mind-set. Being proactive means actively looking for opportunities to enable competitive advantage at the marketplace while also embracing creative thinking among employees to pursue more opportunities. Adaptiveness is described as employees having the ability to perform in multiple roles in projects. Generative ability motivates employees for continuous learning, increasing their competency in multiple areas while sharing knowledge and information in the organization.

Sherehiy et al. (2007) also discussed the agile workforce based on the conceptual models of Dyer and Shafer (2003) and Griffin and Hesketh (2003), such as ability to anticipate requirement change; having self-motivation and creativity abilities; capability to work with individuals in different roles and with different cultures; aptitude for continuous learning; knack for positive attitude towards new technology and changes; capacity to deal with stress in the working environment and having the aptitude to manage unexpected situations.

In software development, developers need to deliver working software with less bugs and performance issues in order to meet customer satisfaction. This approach can be achieved with educating the development teams to have high and diverse skillsets, the ability to use new technologies, the ability to self-adapt and respond to unexpected changes (Youndt et al., 1996). In a related study, Li (2012) identified that as an individual, a developer needed to have development ability, learning ability, ability to predict changes, and versatility in terms of skills to provide agility in the enterprise information system. He also mentioned that development teams need to have experience, internal communication, flexibility and customer initiatives.

Janz (1999) investigated the impact of self-directed work teams on an organization's performance from 231 IS professionals with 28 different SDWT systems

developments within 13 organizations. Based on the result, he found that cooperative learning levels played the most important role affecting work outcomes of an organization.

In addition, Lagerberg et al. (2013) conducted an empirical study on a large-scale software development to investigate the impact of agile practices and principles in global telecommunication company, Ericsson, with two different projects. The first project consisted of 420 team members from 15 teams with the characteristic of separating the systems analysis, design and integration stages and only delivering product once every nine months, while the second project consisted of 120 team members from 14 teams with the characteristics of a cross-functional team delivering software product in three weeks. Each cross-functional team in the second project at least must have developers with roles of system analysts, designers and testers, product owners and scrum masters.

Many developers can remain working effectively in small teams, but it is quite difficult to achieve agility when developers are involved in large-scale software development. Moore and Spens (2008) found that continuous integration and dependency management were the main challenges on a large-scale project within Siemens Medical Solutions which acted as one of the leading companies in the healthcare industry in the USA. The project faced challenges when team members needed to put more effort into cross-functional meetings. The project consisted of 300 people with different types of roles such as product analysts, developers, testers and scrum masters with a team consisting of 25 team members with the details of one scrum master, four to five pairs of developers, one to two analysts and testers (Moore & Spens, 2008).

2.2 Competencies

Business competencies refers to the ability of an organization to develop unique business practices and services or products that makes it hard for competitors to copy (Yusuf et al., 1999). Hence, competent managers and employees are required to learn and respond quickly to changes and market opportunities, including competitive threats (Assen, 2000).

According to Quélin (2000), organizations must encourage learning and understanding of work practices organization-wide by implementing and managing the competencies required. In addition, employees must be encouraged to interact with each other, evaluate their performances, including their ability to know how to act (knowledge

in practice) and being able to act (adaptation of tools and practices that are available in the organization).

Structured organizations lack a wide range of competencies with their individual employees since competencies are limited to one or two tasks, restricting innovative solutions (Siggelkow & Levinthal, 2005). According to Siggelkow and Levinthal (2005), agile organizations come up with diverse solutions providing long-term positive impacts on an organization's performance. Agile organizations set out to develop a set of competencies to support their business strategies and have a horizontal structure to accumulate resources, manage business strategies and foster innovation (Quélin, 2000).

According to (Assen, 2000; Hopp & Spearman, 2011), technology is a great enabler for organizations to have a wide-range of competencies with their employees. The employees must have skills to work with certain technologies, including the integration of technologies to support employee tasks and duties.

Decision-making requires individuals with wider competencies at all the levels in business organizations, and requires managers to be aware of product performance at the marketplace and helping organizations to achieved strategic goals (Sharifi & Zhang, 1999).

Breu et al. (2002) identified the competency of "speed" as vital to swiftly being able to develop new skills, adapt to new changes, learn IT skills (hardware and software) and to develop management skills to achieve workforce agility.

Semeijn, Van Der Heijden, and Van Der Lee (2014) investigated managerial competencies based on multiple rating criteria to identify managerial competencies. Other managerial competencies are the ability to drive integration, plan and manage change management, to ensure operational effectiveness and efficiency, provide highly innovative products and empowered employees, and adopt high productivity technology (Sharifi & Zhang, 1999).

The agile practices in software development emphasize that competency is very important to enable development teams to deliver working software in fast cycles (Misra, Kumar, & Kumar, 2009). Chesebrough and Davis (1983) highlighted the recommendation for IT professionals to become more responsible for planning their own professional careers.

Turley and Bieman (1995) conducted a qualitative study to identify competencies that could differentiate software developers into two categories: exceptional and non-

exceptional. The authors involved 129 developers at one software company and found that exceptional developers had diversity of action, could maintain the big picture of software projects, played proactive roles, had actions driven by a sense of mission and discussed with other developers if problems were encountered in software projects.

In a related study, Colomo-Palacios, Tovar-Caro, García-Crespo, and Gómez-Berbís (2012) identified the technical competencies of IT professionals from one of the leading software development companies in Spain involving 50 IT professionals. Based on their findings, software quality is the most important competency followed by tools and methods in software engineering and software testing, while software configuration management and software maintenance were considered less important.

Based on research conducted by Lindvall et al. (2002), they mentioned that a software project needed at least 25% competent developers with high experience. As a result, the competencies of agile development teams not only concern the pace of software development, but also concern delivering working software according to the requirements of customers and being able to deal with changes (Misra et al., 2009).

2.3 Speed

Speed in agile organizations refers to the organization's capability to respond swiftly to unexpected requests for change from customers and to be able to provide competitive advantage (Gandossy, 2003). Sharifi and Zhang (1999) defined the term "quickness" (speed) as the organization's capability to execute business operations rapidly in delivering products and services to customers including the timely delivery of recent product developments by organizations to the market.

Agile organizations have to possess speed in providing solutions to short-term problems including plan strategies to achieve long-term benefits (Chonko & Jones, 2005). According to Vesey (1992), speed-to-market enables better market opportunities, requiring to embrace innovation and utilize proven tools. When competitors are trying sell their products, agile organizations are already marketing new products. If competitors provide products with the similar cost and quality, agile organizations have the capability of "time-to-respond" as a deciding factor (Chonko & Jones, 2005). According to Shapiro (2001) time management is important for organizations to develop a product that can satisfy customer demand, allowing the organizations to get a head start on the competition.

Zander and Kogut (1995) identify the importance of speed in knowledge transfer within organizations to obtain competitive advantage at the marketplace. They also mentioned that speed could improve the capability to develop innovative product. According to Talaulicar, Grundei, and Werder (2005), speed in decision making is critical for organizations to grow rapidly in a competitive environment. When the organization is dealing with change, the employees must be flexible and open-minded, utilizing organizational resources to provide working solutions to solve customer problems (Chonko & Jones, 2005).

In agile manufacturing, speed is very important with the production process because it allows organizations to meet their customer demands in their supply chain distribution (Allwood et al., 2016). Kessler and Chakrabarti (1996) developed a conceptual framework that captures the relationship of innovation speed and production outcomes. It shows that development speed is highly critical when organizations are in dynamic environments, impacting an organization's capability to achieve strategic outcomes. Speed also impacts project success since this may have an impact on cost of product development and product quality.

A success factor for a software project is on-time delivery of the software products to the market, requiring highly productive development teams. Being late at the marketplace often results in the loss of both current and potential customers (Blackburn, Scudder, & Wassenhove, 1996). Blackburn et al. (1996) identified eleven factors that can help development teams to reduce the overall time for a software project:

1. The use of prototyping
2. Gaining a thorough understanding of customer requirements
3. The use of case tools
4. Parallel development of features
5. Improving the quality of features
6. Improved project team management
7. Better testing strategies
8. Reuse of codes or modules
9. Changes in module size
10. Improvement in communication between team members
11. Having programmers with high skills

Herbsleb and Mockus (2003), conducted an empirical investigation on communication and speed with vendors in the global software development environment.

They identified strategies to speed up collaboration in a global development environment through better utilization of resources, technology and practices to provide communication between different sites by increase in the use of informal communication, splitting the work across sites optimally, awareness and finding experts (Herbsleb & Mockus, 2003).

Tonelli et al. (2013) investigated whether the effect of agile practices could improve the delivery time of software products based on their investigation of a team which included 109 professionals engaged in different aspects of software development. The authors stated that the professionals preferred to adopt various agile practices from different methods than from a single agile method for software development. In addition, customer satisfaction was one of the determinant variables for agile practices to improve delivery time of software development which also involved professionals' perception.

2.4 Responsiveness

Responsiveness means having the organizational capability to detect, anticipate and deal with changes (Sharifi & Zhang, 1999). Through responsiveness capability, agile organizations survive volatile markets (Christopher, 2000). In addition, responsiveness ability enables agile organizations to reconfigure their processes to meet new market requirements, share information organization-wide and adopt new technologies to create innovative product, enabling competitive advantage (Hoyt, Huq, & Kreiser, 2007).

Structured organizations are usually less responsive to customer demand because managers and employees may not have the authority to make decisions when interacting with their customers (Moon, 1999). In structured organizations, often employees are not allowed to take part in high risk activities which potentially limits the information flow between low-level units, sub-groups and executive levels (Moon, 1999).

Sufian and Monideepa (2013) developed a model to examine an organization's responsiveness and performance abilities with a business partner (supplier). Hoyt et al. (2007) identified five elements to measure organizational responsiveness: environmental scanning, strategic planning, flexible manufacturing infrastructures, supply chain governance mechanism, and multi-skilled workers. Hence, organizations can now evaluate their agility capability to response to rapid changes in the market.

Suppliers and partners integrating their systems could enable organizations to achieve more responsiveness to their customer demands (Christopher, 2000). In a related study, Billington and Amaral (1999) indicated that sharing information could improve an organization's responsiveness behaviour although the impact of sharing information is more less compared to the effect of delayed configuration.

In a large-scale software development, it is challenging for development teams to meet their delivery targets while also being responsive to new and extra feature requests. Most software companies prefer to focus on the scale of a software project because customers often ask for features that add complexity to the version and may interfere with the software project (Olsson, Sandberg, Bosch, & Alahyari, 2014). Hsu, Chan, Liu, and Chen (2008) investigated user feedback to learn if it improved software quality and reduced uncertainties. They found that customer involvement during the development phase actually reduces the requirement uncertainty rather than them just being involved once during the requirement phase.

Nidumolu (1995), conducted an empirical study to investigate the impact of uncertainty, and vertical and horizontal coordination on software project performance. His study involved 64 software development projects from the banking industry. His study findings showed that vertical coordination reduces the risks and uncertainty in a project while horizontal coordination had a positive effect on project performance.

Olsson et al. (2014) identified three approaches that could help software development teams to achieve responsiveness capability based on his case study research with Ericsson: traditional development using scale, customized development and a combination of responsiveness and scale. They also suggested several practices such as the use of cross-functional teams, effective allocation and utilization of resources, shortening the feedback loops, quality driven feature development, mind-sets for new features and having the mentality for continuous improvement.

2.5 Knowledge Management

Knowledge management has become vital and organizations must control the flow and integrate information to create a knowledge base for innovation, development, testing and marketing, including for technical research (Pérez-Bustamante, 1999). Knowledge management refers to the process of transferring important information so that there is organizational capability to organize and have proficiency to carry out the tasks, have

dynamic learning abilities, create sustainable plans and have problem-solving abilities (Gupta, Iyer, & Aronson, 2000).

Knowledge management in structured organizations allows only vertical transfer of knowledge which is from senior staff to employees they manage, while cross-functional collaboration is required to allow horizontal knowledge transfer (Walczak, 2005). Hence, structured setups reduce the opportunity for individual growth, restrict well thought-out solutions due to lack of interaction and mostly rely on explicit forms to transfer knowledge within organization (Mahmoudsalehi, Moradkhannejad, & Safari, 2012).

Dove (1999) explained the importance of effective knowledge management in an agile enterprise which is required to have a balanced approach between knowledge management and change proficiency. According to Dove (1999), knowledge could be measured through its strategy and competency, including how people utilize the knowledge and knowledge repositories. In addition, change-proficiency focuses on the ability of the organization to integrate people, processes, practices and procedures to respond to the changes with various organizational competency (low cost, flexible, rapid and low expense). Hence, knowledge management is an important long term task because the organizational approach for knowledge management may affect the quality of service being provided and organizational resources (Darroch, 2005).

In an agile organization, knowledge management served as the basic foundation that encouraged team members to exchange ideas within the organization which leads to innovation, insight and intellectual capital (Chandani, Neeraja, & Sreedevi, 2007). Intellectual capital in knowledge management enables the agile organization to anticipate emerging needs, satisfy current needs, and remove unnecessary needs mostly in the form of tacit knowledge within the organization rather than that kept in the form of documents or stored in online repositories (Dove, 1999).

Pérez-Bustamante (1999), suggests that organizations must implement a culture for knowledge management to identify and implement innovations. The most important aspect in knowledge management is that employees at all levels must help to create, share, protect and improve knowledge repositories in their organizations. According to Wiig (1997), for an enterprise to maximize benefits from knowledge management, the following four aspects are important:

a) Organizations must have an effective approach (operational) to obtain knowledge to build, manage and update their knowledge assets.
b) Organizational staff must have the responsibility to create and organize the infrastructure of knowledge management.
c) Organizations must also have governance procedure to monitor and control their knowledge base, including encouraging employees towards knowledge management activities.
d) Organization-wide awareness towards the value of knowledge though knowledge distribution and utilization from their knowledge repositories.

According to Rouse (2007), there are three ways to categorize the value of information in agile setups:

1) Provide ways for employees to pursue their motives (usefulness)
2) Provide easy access and use of knowledge repositories (usability)
3) Support teams to achieve short-term plans (urgency)

The knowledge management strategy allows organizations to add value to their business and share knowledge among their employees through socialization enabling the capture, disseminating and internalizing of knowledge (Gupta et al., 2000). The reason behind this is knowledge management is considered as one of the processes that cover most of the activities in an organization and there are so many problems from past management (Wiig, 1997).

One of the key activities in software development is logging where developers need to obtain the knowledge (Dávideková & Ml, 2016). The reason behind this is that when developers log information, it requires a large repository which can also easily become flooded with unnecessary information.

Xue-Mei, Guochang, Yong-Po, and Ji (2009) investigated a knowledge management system for software testing. They proposed a knowledge management system based on an ontology to solve problems such as low usage rate, challenges for knowledge transfer, poor sharing environment and inability to achieve optimal distribution of human resources. Chandani et al. (2007) showed that knowledge management could help developers to improve execution and coordination in projects, enhancing delivery speed.

2.6 Organizational Learning

Organizational learning refers to the process of building, supplementing and organizing knowledge around their activities to improve efficiency and adaptation (Dodgson, 1993). Organizational learning provides the organizational ability to innovate through sharing the vision and developing innovations through initiatives (Yeung, 1999). Organizational learning initiates with individual action and then moves to the group level followed by the organization level, forming a continuous learning process involving knowledge acquisition, transfer and integration (Jerez-Gómez, Céspedes-Lorente, & Valle-Cabrera, 2005).

Structured organizations with a centralized structure reinforce their employees to perform single-loop learning. This type of organizational learning happens when the organization changes its strategies to satisfy organizational performance after an error, resulting in no organizational change to the strategies as long as organizational performance is within an acceptable level (Shrivastava, 1983).

On the other hand, agile organizations adopt double-loop learning as their learning process. Double-loop learning enables the organization to produce different results and outcomes compared to solo-loop learning, through changes in organizational strategy and procedures (Pérez-Bustamante, 1999; Shrivastava, 1983). Agile organizations evaluate current activities to become better, adapting themselves to be competitive (Pérez-Bustamante, 1999). Jerez-Gómez et al. (2005), highlighted the importance of organizational learning in a competitive environment and developing a measurement scale that could be used by managers to transform their company into a learning organization.

Jerez-Gómez et al. (2005) proposed a model for managers to measure the learning capability of their organizations based on their investigation of 111 Spanish firms in the chemical industry. They identified four dimensions that affected learning capability in organizations: adaptive management, systems perspective, knowledge transfer and integration. The major benefits of organizational learning are to develop and understand various approaches for employees to avoid the same mistake, the parallel development between learning organizations and individuals, creation of competitive advantage, enhancement in competence and capacity of workforce for competitiveness and changes (Appelbaum & Gallagher, 2000).

Marquardt (1996) elaborated that there are three optimal dimensions for employees to create a dynamic learning environment: (1) levels of learning (individual and group learning), (2) different types of learning (active, anticipatory, flexible and reflective learning), and (3) organizational learning skills (communication, shared vision, systems thinking, specialization, team learning and concept visualization). In another study, Yeung (1999) identified four learning styles: (1) continuous improvement, (2) experimentation, (3) benchmarking and (4) competency acquisition.

Antonacopoulou and Chiva (2007) investigated the complexity of organizational learning based on complexity procedures – schema variety and interaction interdependence. Their study shows that organizational learning could be described as a learning process between multiple employees engaged in organizational work, if necessary adjusting the learning process in order to encourage employees from different functional areas to work together.

Software development based on a highly collaborative environment requires an intensive learning process involving different roles and a large number of developers (Ras & Weber, 2009). Arechavala-Vargas, Diaz-Perez, Madrigal-Torres, and Ferrer-Ramirez (2007) explained that the processes of organizational learning are constructed from collaboration and trust among individuals to solve problems. They also categorized organizational learning into four different levels based on organization maturity, where:

1) The first level focuses on problems that cannot be handled by the organization,
2) The second level can be obtained through evaluation based on previous decisions,
3) The third level is derived when the decisions satisfy the performance evaluation in order to achieve the organization's goals,
4) And the fourth level can be found when the organizational learning process involves two or more organizations.

Their findings showed that the role of the early culture of a software company could lead to an effective learning and capital building process. Huntley (2003) focused on the impact of organizational learning on open-source programming projects and identified three key points related to organizational learning which involve the system complexity, adaptive learning and learning curve effects. Adaptive learning is defined as the approach of development teams to evolve and respond to opportunities, problems and changes in the working environment. The process of adaptive learning is applied during the debugging process to help developers overcome software complexity and provide feedback about the software's quality (Huntley, 2003).

2.7 Organizational Culture

Organizational culture plays an important role shaping the behaviour of employees. According to Schein (2010), there are three levels of organizational culture:

(1) At the surface level, it shows the visible part of the organizational culture (norms or procedures).
(2) At the middle level, it covers beliefs and values of what should be done within organizations.
(3) At the deep level, employees are completely aware of the behaviour of other employees without observing them (shared assumptions) (Iivari & Iivari, 2011; Schein, 2010).

There are seven main characteristics of organizational culture identified by O'Reilly, Chatman, and Caldwell (1991): people orientation, team orientation, innovation and risk, detail orientation, outcome orientation, aggressiveness and stability.

The organizational culture in structured organizations is based on the assumption that power should be with the top management, dependent upon predetermined roles and responsibilities in order to pursue organizational goals and stakeholders' interests (Janićijević, 2013). Employees follow the procedures to coordinate tasks with other employees from other departments where each process has written procedures, regulations and manuals to be followed to the full extent (Janićijević, 2013).

Agile organizations adopt a collective approach, have a well-coordinated action for efficiency and are well prepared to face the market with rapid changing requirements (Janićijević, 2013; Van Veelen, Storms, & van Aart, 2006). In an agile environment, employees work together and have mutual benefits without considering individual goals to provide a significant contribution to the activities. Barney (1986) explained that organizations must have the three characteristics of value, uniqueness, and flexibility to engage in a competitive environment. These organizational characteristics help to improve the performance with their own approach, and without an organizational culture, organizations could not obtain a competitive advantage (Barney, 1986).

Iivari and Iivari (2011) investigated the correlation between organizational culture with the deployment of agile methods by using the Competing Values Framework (Quinn & Rohrbaugh, 1983) for organizational culture where enterprises could achieve agility (developmental culture) when they embraced change and focused more on external

factors rather than operational ones. They found that agile methods were not compatible with an organization with a relatively strong hierarchical structure.

Gotwon and Ditomaso (1992) also investigated the relationship between two factors of organizational culture that could affect an organization's performance, which was measured based on the responses from participants related to stability and adaptability. Based on their findings, an adaptability culture provided better insights and predicted the organization's short-term performance. According to Sherehiy et al. (2007), the culture of change in organizations starts from a supportive environment that encourages employees to perform continuous improvement through training and positive attitudes in dealing with changes, ideas, people and technology.

The organizational culture presents as the way for acting and thinking by employees and it can be seen as an important value that holds organizations together (Passos, Mendon, & Cruzes, 2014).

Tolfo and Wazlawick (2008) investigated the aspects of an organizational culture that might influence the output from Extreme Programming (XP) practices from six software companies. From their study, organizational culture could facilitate or constrain the adoption of the XP method because not every software development teams would adopt the same organizational culture. They developed the organizational culture dimensions from O'Reilly et al. (1991) framework. However, if the adoption of XP practices can be managed properly through an organizational culture, developers can produce good results in software projects (Tolfo & Wazlawick, 2008).

Verma and Amin (2010) proposed a conceptual framework to identify the risk management from software practices by using several cultural factors that might become major causes of software failures.

Passos et al. (2014), conducted an empirical study on four software companies in Brazil to investigate their organizational culture in terms of origins, sources and impacts on software development practices. The authors developed a conceptual framework from the Theory of Reasoned Action and found new insight that can be adopted by developers. The findings can be categorized into three levels of culture (values, assumptions and artifacts) and related to the relationship between software practices and organizational culture. Practices identified based on the agile approach relating to agile culture are face-to-face meetings between representatives from the project team and the customer where

aim is to get clarification about requirements and enable project teams to focus on the most important features (Verma & Amin, 2010).

2.8 Cooperative Teams (Team Effort)

Agile organizations are strongly driven by internal and external cooperation (Goldman, 1995). Cooperative teams have all their individuals working collectively on a task. Brown and Palincsar (1989) suggested that in a cooperative environment, team members collectively provide the best analysis and ideas for problem solving, leading to better outcomes.

Structured organizations employ specialists and differentiate responsibilities of each employee in a formal way, which means that every task is assigned to an individual rather than to a team where employees focus on their assigned work (Moch & Morse, 1977), whereas in the agile organization, employees are assigned in small teams and not only represent their functional departments but become committed members to the whole-team effort concept (Crispin & Gregory, 2008). A task in an agile team is often assigned to two or more people, working together towards a common goal, which also prevents them focussing on individual gains (Yauch, 2007).

Agile organizations require cooperation and flexibility organization-wide in order to respond to customers' needs (Maskell, 2001). According to Nerur, Mahapatra, and Mangalaraj (2005), cooperative social processes were required in order for agile organizations to become successful based on communication and collaboration based on value and trust. The cooperative and flexibility abilities are also extended to third parties such as customers and suppliers to develop value-added products and services (Maskell, 2001).

Beersma et al. (2003) examined the impact of team performance and reward systems on team composition, task dimension and individual performance. Their findings suggested that a competitive environment encouraged the individuals to achieve speed while a cooperative environment helped to achieve accuracy. According to Beersma et al. (2003), the impact of a reward system could work effectively in a team environment where performance level is below than what is the expected.

Kohn (1999) argues for a collaborative environment but without the reward system that promotes competition affecting team performance. Janis and Mann (1977)

highlighted that the negative effect of excessive cooperation could prevent conflicts in the team and individuals prefer to exclude their opinions during important discussions. Although conflict could lead to work disruption, it is becoming a characteristic of organizations with diverse employees to require them to be flexible and have specialized skills to deliver innovation (Yauch, 2007).

A cooperative concept in agile software development can be identified through self-organizing teams. The objective of self-organizing teams is to work together and participate in team decision making for common goals. Self-organizing teams in agile software development have leadership roles to provide direction, motivate, obtain resources and align people (Anderson et al., 2003; Cockburn & Highsmith). Self-organizing teams usually have a leadership role such as a scrum master who can perform interdependent jobs, and has authority to create plans, do scheduling and assign tasks to team members including making decisions (Guzzo & Dickson, 1996; Moe et al., 2008).

Moe et al. (2008), investigated the obstacles of introducing self-organizing teams in agile software development. Their study involved professional developers from scrum teams and found that managing responsibilities among individuals was the most critical obstacle that development teams need to handle, including lack of autonomy and specialized skills in teams.

Hoda, Noble, and Marshall (2013), conducted a study over four years on self-organizing roles in agile software development teams, which involved 58 practitioners from 23 software organizations in New Zealand and India. Their study revealed that self-organizing roles could help software development teams manage their responsibilities better in their development teams. In order for self-organizing teams to prepare and face new challenges, they must have mutual trust, shared vision, respect and the ability to re-organize (Cockburn & Highsmith, 2001).

Table 6 provide a summary of agile organization concepts based on literature review.

Table 6. Agile Organization Concepts Identified for This Study and Their Definition

Agile Organization Concepts	Definition
1. Workforce Agility	The ability to learn and adapt skills and practices in a short time frame to act on change for successful product development outcomes.
2. Competencies	Organizational ability to develop unique business and development practices including services/products that make it hard for competitors to copy.
3. Speed	The organization's capability to execute business operations swiftly in delivering products/services to market.
4. Responsiveness	The organization's capability to detect, anticipate and deal with changes at the marketplace.
5. Knowledge Management	The process of creating and maintaining knowledge for continuous improvement and effective effort.
6. Organizational Learning	The process of building, supplementing and organizing knowledge around activities to improve efficiency and adaptation.
7. Organizational Culture	The organizational assumptions, beliefs and values that are shared organization-wide to be successful at the marketplace.
8. Cooperative Teams (Team Effort)	Collect cross-functional effort for decision-making and product development.

Organization learning is the most important agile organization concept. At the agile organization level, organizational learning allows organizations to become more successful and adapt to rapidly changing environments which enable employees to find the best solutions to fulfil the organization's goals and objectives (Giesecke & McNeil, 2004). Organizational learning also promotes employees to unlock individual creativity and knowledge creation with the importance of an adaptive and flexible organization structure (Dasgupta & Gupta, 2009).

At the software development level, organizational learning can help software development teams to capture and maintain necessary knowledge with an incremental approach (Kavitha & Ahmed, 2011). Based on the learning focus, the organization can develop choices on how the speed of development is related to other goals such as the cost, risk, quality and innovative content of the project (Mathiassen & Pries-Heje, 2006).

2.9 The Key Practices in Agile Software Development

Product Backlog

Product backlog consists of a list of prioritized features that are needed in the software product and serve as the source for all requirements for any changes to the final product (Pichler, 2010). The tasks are usually defined in the form of user stories and add each product functionality in small increments (Cobb, 2015). Early development of a product backlog only identifies best-understood requirements and expands as the product try to match the environment (Schwaber & Sutherland, 2012).

When the team gets new ideas and new requests for implementation (change) during the development, it is added to the product backlog so the content in the product backlog continuously evolves; hence it is continuously reviewed, approved and prioritized by the product owner (Cobb, 2015). Product backlogs are usually created based on the value, risk, priority and necessity for features, which explains why higher priority product backlog items are more detailed compared to lower priority items where each product backlog item have a description, priority and estimate (Schwaber & Sutherland, 2012).

Daily Stand Up Meeting

During daily stand up meetings, team members report their progress on work items since the previous meeting to the team and provide their goals for the day including providing suggestions related to their task or any other member's task (Sivanantham, 2012). This meeting is usually time-boxed at fifteen minutes and happen at the same location and time, where team members focus on the information shared with the other members to synchronize their work as a team (Dorairaj, Noble, & Malik, 2012).

Vision Planning

The purpose of vision planning is to identify and make decisions on the new functionalities of the product, to elaborate the specification of high-level requirements, decide on the required resources and plan the releases (Wang, Pfahl, & Raffo, 2008). Initially, product owners identify the requirements with collaboration from the development team, customers, industry and other business functions before proposing the new features to the steering committee for implementation decisions. The process of product vision planning enables the product management team to share and extend the product concepts with their software engineering unit and other business functions to

ensure they work together to achieve desired results (Sliger & Broderick, 2008; Werder, Zobel, & Maedche, 2016).

Development teams need to have a good understanding of their client's business processes, existing systems, operations and future goals in order to understand the business needs of the client, ensuring that the core values are part of the key features of the software product (Q. Wang et al., 2008). A successful vision plan implementation requires an organization-wide commitment (Donaldson & Siegel, 2001).

Release Planning

Release Planning is a continuous activity that enables agile development teams to transform product vision into a list of features with priorities for implementation (Shalloway, Beaver, & Trott, 2009). Release planning cannot be done in a single session and is usually revisited when development team need to update the plan at the end of each sprint (Cobb, 2015). There are some critical items to be considered when teams do release planning such as customer satisfaction and time-to-market, including features which are dependent on other features (Agarwal, Karimpour, & Ruhe, 2014).

When dealing with uncertainty, development teams need to put in a collective effort to learn and solve the problem and uncertainty before including the items in the next release (Cobb, 2015). According to Cobb, the level of tolerance in the release planning is higher compared to sprint planning. Development teams can predict the number of user stories to include in a release by using risk factors, velocity and story estimates (Shore & Warden, 2008). Release planning provides a clear direction and effort estimation for development teams to determine the features to be included in the next release to meet stakeholder expectations and ensure the software project within budget and on schedule (Danesh, 2011).

Collective Code Ownership

With collective code ownership practice, the team itself is responsible for maintaining the code base. Each team member has responsibility to make necessary changes if they encounter coding issues, which means that team individuals can fix problems to improve the quality of the code (Shore & Warden, 2008).

Teamwork is essential for collective code ownership practice to produce good code with high quality (Chromatic, 2003). At the beginning, development teams may focus to produce adequate quality code that can work as expected. The sense of team ownership increases when team members fix coding issues and improve the existing code

with shared solutions (Shore & Warden, 2008). Small and frequent releases are required in order to create an effective environment for collective code ownership, which motivates the members for friendship and collective effort as the code evolves one step at a time from one release to another (Chromatic, 2003).

Pair Programming

In pair programming, two developers work together on a single task from the same space, sharing a single development machine while frequently switching their roles from driver to navigator until the completion of the task (Cockburn & Highsmith, 2001). When in the driver role, the developer has the responsibility to focus on the task details to write the appropriate code and while in the navigator role, the other developer must ensure that the code fulfils the team guidelines for implementing and testing, including carrying out any other quality assurance tasks in parallel with code implementation (Chromatic, 2003).

Pair programming reinforces good programming habits since peer pressure to perform important tasks ensures they are not distracted by other tasks (Shore & Warden, 2008). Pair programming practice requires active participation by both developers in the navigation and driver roles to have a desirable outcome (Hunt, 2006).

Test-Driven Development

Test Driven Development (TDD) means bringing upfront relevant testing and quality assurance activities and carrying them out with development for each short development cycle (Farcic & Garcia, 2015). The main procedure of TDD consists of a few steps from writing the test, running and verifying all tests, writing the implementation code, running automated tests and refactoring (Laranjeiro & Vieira, 2013). The benefits of TDD practice are enhanced productivity for testing and development, fewer defects as a result of test automation, results in lower code maintenance and improved cost efficiency because TDD enables development teams to perform testing faster through automation and continuous integration with the code base through unit, component, system and regression tests (Rico, Sayani, & Sone, 2009).

User Story Acceptance Testing

Acceptance testing covers the high-level tests of business operations and is often used by development teams to ensure the software meets the criteria of business goals (Read, Melnik, & Maurer, 2005). Testing practices in agile software development are significant since they enhance communication and feedback for developers, including providing visibility to customers (Haugset & Hanssen, 2008). The main focus of the acceptance test

is on the story acceptance, feature acceptance, usability, user acceptance and alpha/beta test (Leffingwell, 2010). A user story is complete if it passes its acceptance tests. This test confirms if a user story is complete and functioning as expected.

Developers often use automated tools to implement user acceptance testing, resulting in fewer errors, more safety to make changes in the code, no or less manual testing and improved understanding of how the system works from the developer's perspective (Haugset & Hanssen, 2008).

Regression Testing

The purpose of a regression test is to validate the change, e.g. added a new functionality to the software, to ensure the current code still works as intended and there will not be any failure with new code integration with the previous version of the software (Stamelos, 2007). According to Onoma, Tsai, Poonawala, and Suganuma (1998), regression testing should be used whenever there is change and should be include in the overall approach for software development and maintenance rather than perform in an independent stage.

As project progresses, it becomes difficult to perform manual regression tests because it could be possibly range from hundreds to thousands of tests and the code always keep changing (Cobb, 2015). Automating the regression testing ensures and improves the consistency for software quality since now it frees the quality assurance team to concentrate on the value-adding tasks to enhance the quality of the features and product (Cobb, 2015; Hazzan & Dubinsky, 2009).

Sprint Planning

In Sprint Planning, the scrum team usually discusses about how much work the team will commit for the current sprint cycle (Schiel, 2009). At the initial stage, the product owner and the whole team must have clear agreement about the sprint goals and have a clear understanding on the acceptance criteria for each backlog item (Resnick, Bjork, & de la Maza, 2011).

In the second half of the sprint planning, the scrum team does detailed task planning to clearly define how the team will perform the tasks related to the user stories, including assigning them to team members for implementation (Cobb, 2015; Resnick et al., 2011). When development teams encounter, and are unable to solve, defects during the sprint process, they will prioritize the backlog item to a higher priority for the next sprint session, otherwise the task may become a bottleneck for the whole project (Cobb, 2015).

Refactoring

Refactoring refers to a process of changing the internal structure of the current code without modifying its external behaviour (Fowler, Beck, Brant, Opdyke, & Roberts, 2012). With refactoring, developers can improve the design of the existing code and make the code easier to understand, thus allowing the developers to identify defects and implement new features more quickly (Fowler et al., 2012).

Furthermore, refactoring in short cycles enables developers to enhance maintainability in the long term because it can reduce the work needed to be done later while significantly reducing the cost in the software life-cycle (Nidhra, Dondeti, Katikar, & Tekkali, 2012). However, if the developers perform refactoring with implementation, there could be not enough time left at the end, becoming quite challenging for inexperienced developers because it may lead to complex changes in the refactoring process (Vasileva & Schmedding, 2016).

Coding Standards

Coding standards are a set of guidelines that ensure developers have agreement and consistency when writing the code. In projects with a large number of developers, coding standards become important since developers have their own style but a standard is required for consistency so that later anyone with a minimum effort is able to understand the program logic when reviewing or carry-out maintenance tasks (Fowler et al., 2012).

Coding standards enable developers to communicate effectively through their code, achieved through teamwork involving the whole team to establish clear communication in the form of habits and preferences (Chromatic, 2003). The coding standard practices should be short and easy to remember, requiring participation by all the developers to create team values rather than just identifying the types of code to write (Chromatic, 2003; Kelly, 2015).

Retrospective

The main objective of the Retrospective process is to identify alternative choices and improve the outcomes from the entire team's perspective for the next iteration or project (Kelly, 2008). Retrospective often occurs at the end of the iteration or project and the meeting lasts for an hour with the involvement of the entire team and scrum master (Kelly, 2008; Pollard, 2016). The scrum master has the responsibility to ask each person from the team to identify any ideas or backlog items to initiate, continue or stop (Pollard, 2016).

After a list of ideas has been gathered, team members can discuss and decide which backlog items should be implemented in the next sprint. It is important to deal with this kind of problem at an early stage rather than pushing and delaying them into bigger problems which may lead to failure for the development team (Schiel, 2009).

User Stories

User Stories are the requirements to be implemented and the team uses them to create a plan based on their estimates to implement the software functionality (Cohn, 2004). Developers create and gather user stories by using the following techniques: user interviews, questionnaires, observation and story-writing workshops (Monochristou & Vlachopoulou, 2007).

Each user story has a value stated through story points. Hence, a complex user story that describes a large piece of software functionality is allocated a higher story point value while smaller simple user stories will be allocated lower story point values (Schiel, 2016). Before implementing the user stories, developers provide estimates for each user story. If a user story is estimated to take more than three weeks, developers can divide it into smaller pieces to ensure the stories become much clearer and their scope much more limited (Schiel, 2009).

Open Workspace

In Open Workspace developers share workstations, encouraging face-to-face communication between developers and providing flexibility with organizational change (Schifferstein & Hekkert, 2011). According to Schifferstein and Hekkert (2011), developers may find it hard to work from an open workspace but when used effectively it can provide insights into conversations and immediate face-to face feedback on developer queries and issues, if any, during the day.

The layout depends on the number of people sharing the workspace and ensures that there are no small cubicles, including no boundaries in the team workspace helping the developers to focus and collaborate on tasks (Babar, Brown, & Mistrik, 2013; Schifferstein & Hekkert, 2011).

Burn-Down / Burn-Up Charts

A Burn-down Chart provides information on the work or the user stories left to be done versus the available time, allowing prediction of when all of the work will be completed (Solanki, 2009). The burn-down chart shows the remaining hours, velocity, discontinuous plot and planning plot (Goodpasture, 2015). Development teams use burn-down charts to

track the remaining tasks with the remaining time throughout the working sequence unless specified by the project leader (Cobb, 2015). A burn-down chart is usually discussed by the development team during their team meetings in order to update the information on the plan and estimation (Goodpasture, 2015).

A burn-up chart presents information which relates to the number of user stories the development team has accomplished to keep accumulating the functionality until it is completed (Solanki, 2009). From the accumulated functionalities, development teams can compare the goals of the software project such as the release plan or budget in order to provide clear feedback to the entire team (Cobb, 2015). The benefit of a burn-up chart is that it is easy to understand, which is similar to the traditional chart techniques that display the number of story points achieved in the release (Rubin, 2012). Burn-down and burn-up charts are simple yet powerful tools that can help development teams to provide visibility for each iteration if two charts are combined (Cobb, 2015).

Continuous Integration

The Continuous Integration is a software development practice that enables software development teams to integrate the source code in a common repository (Duvall, Matyas, & Glover, 2007). The main objective of continuous integration is to eliminate software integration issues and have "traceability" (Dömges & Pohl, 1998). Traceability is important in software development since it has multiple purposes such as helping teams to identify improvements and guiding evaluation of their development effort, internal follow-up on implementation, and producing an accurate and consistent audit-trail report on implemented features and tests carried out on one specific version of the software product (Stahl, Hallén, & Bosch, 2017).

Self-Organizing Teams

A self-organizing team is a group of employees that work together for the purpose of accomplishing a common goal to deliver software features and products by taking ownership of the work, reducing their dependency on management (Moreira, 2013). Self-organizing teams have values such as mutual trust, respect, a common focus and the ability to re-organize the team itself to meet new challenges (Grisham & Perry, 2005). Self-organizing teams solve implementation problems through collective responsibility, self-assignment and cross-functionality effort, including continuously seeking improvements (Hoda et al., 2013).

Unit Testing

Unit testing is the basic level of testing involved with software development. Hence, a unit test on an individual function of the software confirms that it performs its function correctly, providing a better way for development teams to manage the integration of functions into a program (Lewis, 2016). Therefore, unit testing is crucial to determine the success of the software development since it provides indications of whether a functional unit satisfies the expected positive behaviour or not. Unit testing also acts as the first stage of testing before developers release the application for user acceptance testing (Khan, 2016).

The table below identifies the agile organization concepts, together with the related agile software development practices based on the understanding acquired through the literature review on both. These two were the basis for developing the survey questions provided in Appendix B.

Table 7. Agile Organization Concepts Related to Agile Software Development Practices

Agile organization concepts	Agile software development practices
1. Workforce Agility	• Self-organizing teams • Daily stand-up meeting • Sprint planning • Retrospective • User stories
2. Competencies	• Continuous integration • Self-organizing teams • Product backlog • Test-driven development • User stories • Unit testing • User story acceptance testing
3. Speed	• Sprint planning • Product backlog • Test-driven development • Continuous integration • Refactoring
4. Responsiveness	• Product backlog • Release planning • Sprint planning • Retrospective • User story acceptance testing
5. Knowledge Management	• Pair programming

	• Test-driven development • Retrospective • User story acceptance testing • Regression testing • Refactoring • Coding standard
6. Organizational Learning	• Vision planning • Retrospective • Release planning • Sprint planning • Retrospective • Test-driven development
7. Organizational Culture	• Open workspace • Coding standard • Collective code ownership • Retrospective • User story acceptance testing
8. Cooperative Teams	• Vision planning • Product backlog • Daily stand-up meeting • Pair programming • Sprint planning • Coding standard • Retrospective

Chapter Three: Hypothesis

3.0 Knowledge Management and Organizational Learning

Knowledge management refers to an organization's capability to organize, transfer and integrate to create a new knowledge base required to create achievable work plans and the ability to learn on the fly (dynamic learning) to accomplish tasks (Gupta et al., 2000; Pérez-Bustamante, 1999). The core value of knowledge management is facilitating team members to collaborate, communicate, coordinate and share knowledge to successfully accomplish a task (Duffy, 2000).

An agile organization has processes to generate, build and organize knowledge around their daily activities and continuously develop through innovation in order to adapt and have organizational efficiency (Dodgson, 1993; Yeung, 1999). The organization must go through knowing, understanding, thinking and learning to become a learning organization. An agile organization encourages its employees to adopt the process of double-loop learning which allows the organization to produce better results through changes in organizational strategy and procedures (Pérez-Bustamante, 1999; Shrivastava, 1983). The major benefits of organizational learning are developing and understanding various approaches for employees to avoid the same mistake, the parallel development between learning organizations and individuals, creation of competitive advantage and enhancement in the competence and capacity of the workforce for competitiveness and change (Appelbaum & Gallagher, 2000).

In agile software development, organizational learning refers to the capabilities of the organization to survive, develop and increase innovation (Arechavala-Vargas et al., 2007). Organizational learning in agile software development serves as a critical measure of performance in the project (Landaeta, Viscardi, & Tolk, 2011). Based on the learning focus, the organization can develop choices on how the speed of development is related to other goals such as the cost, risk, quality and innovative content of the project (Mathiassen & Pries-Heje, 2006). All agile software development practices, tasks and activities require solid knowledge management and continuous learning to undertake these tasks to be able to successfully deliver any project.

When two developers work to design and test software together, they obtain various kinds of knowledge in forms of task-related knowledge (systems knowledge, coding invention, design practices, technology knowledge and tool usage tricks) and contextual knowledge (using past experiences to determine whether it is appropriate to

use specific design patterns in different coding projects) (Sivanantham, 2012). It is recommended that project teams perform pair rotation often in order to maximize the use of pair programming and ensure the knowledge is shared among team members (Cobb, 2015).

Reflection encourages development teams to review their tasks, reflect on how to make the activities more efficient and adjust the whole process to fit in with the current situation (Unhelkar, 2016). According to Talby, Hazzan, Dubinsky, and Keren (2006), reflection is one of the beneficial ways for development teams to enhance collaboration and team effectiveness because each member may work and act differently towards output in the long term. During this process, it is very important to convert tacit knowledge to explicit knowledge and put the knowledge in common areas so everyone can access the resources (Sivanantham, 2012). Development teams can also use reflection as a mechanism to conduct external communication with customers in order to gain significant feedback that can be useful for project goals (Pikkarainen, Haikara, Salo, Abrahamsson, & Still, 2008). Using reflection can create awareness for software developers to improve their skillset because the area becomes very broad which means required skills that can work successfully in certain areas from the past (e.g., 30 years ago) may not apply in the current practice (Capretz, 2003).

When developers adopt the Test-Driven Development (TDD) approach, they write detailed specifications effectively by using a Just-in Time (JIT) basis (Sivanantham, 2012). Developers need to understand about user requirements and obtain the required information before writing test cases in a TDD environment (Farcic & Garcia, 2015; Laranjeiro & Vieira, 2013). The reason behind this is when developers develop test cases without understanding user requirements, they will end-up writing unnecessary test cases which are not efficient for the software project (Rico et al., 2009). Using TDD, developers can specify the requirements and validate their work from writing the tests that aim either at developer level or acceptance level before writing the actual code to fulfil the tests (Sivanantham, 2012).

Developers could improve their learning process and execution and coordination between internal team members and enhance speed to delivery and accurate estimation for the software project with organizing, transferring and integrating the knowledge.

Hypothesis 1: Knowledge management positively affects the organizational learning of agile software development teams or organizations.

3.1 Organizational Culture and Organizational Learning

Organizational culture in agile organizations is based on coordination, collaboration and informal communication to enable mutual benefits that provide competitive advantage at the marketplace (Janićijević, 2013; Van Veelen et al., 2006). Without the shared agile culture organization-wide that is valuable, unique and flexible, organizations will have less opportunity to obtain competitive advantage (Barney, 1986). The major benefit of the agile culture is continuous learning by employees to improve the organization's performance by adapting to changing situations, decision making, embracing change for better results, encouraging cooperation and continuously getting highly valued feedback on their work (Gotwon & Ditomaso, 1992; Iivari & Iivari, 2011; Sherehiy et al., 2007). The agile culture helps to deliver competitive products by identifying and producing customer-oriented products, teamwork, learning and a mind-set for change (Sherehiy et al., 2007).

The entire agile development team must communicate, collaborate and work together through intensive interaction to identify and deal with implementation issues and challenges for successful implementation (Highsmith, 2002; Wendorff, 2002). The collective effort is critical to deal with any client request for changes requiring the entire team for product backlog meetings for discussion, re-evaluating and setting new priorities to tasks in order to deliver the software on time (Cobb, 2015; Schwaber & Sutherland, 2012).

According to Lindvall et al. (2002), rapid interaction and communication, accepting requirement changes, trusting people, collective team effort and fast feedback (learning) from customers is the key agile culture.

In software development teams, the agile organizational culture is enforced through an open workspace for the entire team. An open workspace also encourages learning on the collective effort for the entire team by sharing information and having spontaneous interaction and immediate feedback between developers (Schifferstein & Hekkert, 2011). The agile team consists of team members of diverse backgrounds and different functional departments to enable effective collaboration including often adopting pair programming practices (Babar et al., 2013). The on-site customer (business oriented) role which provides cross-functionality in agile development teams is critical for decision making, team learning coordination, collaboration and informal communication (Martin, Noble, & Biddle, 2003).

In agile software development, pair programming has a positive impact on scheduling of tasks, acquaintance culture and team cohesion (Vanhanen & Lassenius, 2007). Pair programming enforces the agile culture for learning to improve the engineers' discipline, helps to implement better solutions, grows team morale and enhances knowledge required to carry out work on projects (Nosek, 1998; Williams, McDowell, Nagappan, Fernald, & Werner, 2003). These benefits of pair programming lead to improvement in the quality of the software (DeClue, 2003; Hanks, McDowell, Draper, & Krnjajic, 2004; McDowell, Werner, Bullock, & Fernald, 2002).

Organization learning enables the entire team including the scrum master discusses and agrees on which tasks should be concentrated during sprint planning with the sprint duration between two weeks and four weeks (Schiel, 2009). As the team becomes more mature, they learn from their actions and act on the learning (Cobb, 2015). The development team can perform low priority tasks with flexibility while crucial tasks must be completed in the same sprint cycle to avoid the introduction of new problems for the next iteration (Cobb, 2015), otherwise the development team needs to adjust their sprint cycle at a later stage of the development or possibly hire an agile coach to help them solve the problem (Srinivasan & Lundqvist, 2009).

After the development team reaches the end of a sprint cycle, they often perform a sprint retrospective (sprint review) meeting to continuously increase awareness and knowledge from multiple perspectives to deal with certain problems during software development (Kelly, 2008; Pollard, 2016). Collaboration from the entire team can produce higher organizational learning levels which lead to better performance (Arechavala-Vargas et al., 2007). The sprint review process becomes more effective for the overall learning process when the development team gains a better understanding of the process (Srinivasan & Lundqvist, 2009).

Another example that reflects organizational learning in software development is TDD. In a TDD environment, developers build and test the code at the same time for each new functionality that they want to implement in several repetition cycles before writing the actual implementation code (Farcic & Garcia, 2015). The TDD process may represent an overlap with the tester role, however it can be considered as positive aspect of the learning process because the developer ends up implementing the test during software development and it can improve the test estimation for the software project (Santos, Goldman, Shinoda, & Fischer, 2011).

As a result, an organizational culture encourages the entire team to learn, interact and collaborate, leading to delivery of software projects.

Hypothesis 2: Organizational culture positively affects the Organizational Learning of software development teams or organizations.

3.2 Organizational Learning and Competencies

Organizational competencies deal with the ability of an organization to develop unique product development practices and have the ability to respond quickly to market opportunities, including having unique capabilities to deal with competitive threats (Assen, 2000; Yusuf et al., 1999). The major benefits of competency are to establish a strategic product vision, continuous improvement in product quality, cost and operations; effectiveness / efficiency; achieving effective internal and external collaboration and having in-house ability to manage change and empower employees (Sharifi & Zhang, 1999). In order for employees to become competent they must have a swift learning ability to develop new skills, obtain necessary skills to perform changes, acquire innovative management skills and obtain technical skills (Breu et al., 2002).

An agile organization encourages their employees to adopt the process of double-loop learning which allows the organization to produce better results through changes in the organizational strategy and procedures (Pérez-Bustamante, 1999; Shrivastava, 1983). The major benefits of organizational learning are to develop and understand various approaches for the employees to avoid the same mistake, the parallel development between learning organizations and individuals, creation of competitive advantage, enhancement in competence and capacity of workforce for competitiveness and changes (Appelbaum & Gallagher, 2000). In agile software development, learning is crucial for development teams to meet their development targets and increase innovation (Arechavala-Vargas et al., 2007).

Competencies facilitate developers to play proactive roles, carry-out diverse tasks and actions, maintain the pace of software development and deliver the software according to customer requirements (Lindvall et al., 2002; Misra et al., 2009). Without competence, this can pose serious problems such as affecting the morale of the team (Nerur et al., 2005).

Agile software development requires solid competencies on incremental development, continuous integration and test driven development (unit test, integration test, alpha/beta test, story acceptance, feature acceptance, usability and user acceptance test) (Humble & Farley, 2010; Laukkanen, Paasivaara, & Arvonen, 2015; Leffingwell, 2010). Developers require competencies to be able to perform automation during the acceptance testing phase in order to reduce the effort when customers change the requirements so they can concentrate with other critical tasks (Haugset & Stalhane, 2012).

Agile software development teams must consist of competent individuals so that they can step into different formal and informal roles required for the teamwork (technical leader, team leader, project manager, developer and tester) (Hedberg, 2015).

The organization has enough software competency when development teams can perform the tasks in coordinated actions (Barton & Brooks, 2012). Agile teams also encourage team members to work in different types of projects when necessary (Sundararajan, Bhasi, & Vijayaraghavan, 2014).

Agile software development is dependent upon the competency developed through continuous and swift learning abilities of the individuals and the team as a collective unit.

Hypothesis 3: Organizational Learning positively affects the Competencies of software development teams or organizations.

3.3 Organizational Learning and Responsiveness

In an agile organization, responsiveness capability is described as the ability of an organization to anticipate, detect and deal with changes including ability to swiftly plan that allows for an immediate response to deal with the threat (Sharifi & Zhang, 1999). Responsive behaviour is critical to be able to swiftly adjust business objectives and goals and deliver products according to market requirements (Breu et al., 2002). Agile organizations have responsive behaviour to swiftly determine and deliver market-driven products before competitors (Kusiak & He, 1997).

The agile workforce must develop responsiveness attributes through learning and empowerment, allowing them quickly to adapt to the changed work environment which requires new skills development, having workplace independence (virtual teams) and requiring responsive ability to deal with any external change (Breu et al., 2002).

The major advantage of having responsiveness capability by agile organizations is that they are swiftly able to identify and share information organization-wide on changes in the marketplace, while being able to re-configure processes and practices to meet new market requirements and obtain an early competitive advantage (Hoyt et al., 2007).

Responsive behaviour requires the software development teams and organizations to have the ability to swiftly learn, adapt and respond to deliver to market needs (Salo & Abrahamsson, 2005). Agile software development teams require responsiveness capability in order to deal with, or create, change and respond rapidly to changes from uncertain environments (Cockburn & Highsmith, 2001). Responsiveness capability means agile software teams and organizations have the ability to modify their processes, practices and structures in a turbulent environment (Reed & Blunsdon, 1998). In an agile software development environment, the customer requirements swiftly change, requiring product creativity and innovations (Vazquez-Bustelo, Avella, & Fernández, 2007). A critical part of this behaviour for agile in-house software development is learning practices to swiftly acquire new knowledge, skills and support to deliver market-driven products.

Boehm (2002) identified the requirement for highly skilled developers, customer involvement, architecture, refactoring and size of a development team as crucial for having responsiveness capability. According to Law and Learn (2005), responsive behaviour can be achieved through adopting pair programming to up-skill newcomers. To be agile in a market-driven environment requires teamwork response (this is a key part of the core competency to continuous enhancement though learning (Vernadat, 1999). Agile software development requires an on-going organizational reconfiguration in structure, strategies, roles and management style.

Hypothesis 4: Organizational Learning positively affects the Responsiveness Behaviour of agile software development teams or organizations.

3.4 Organizational Learning and Speed

In an agile organization, speed capability enables them to swiftly deliver products including ability to swiftly respond to unexpected changes to provide competitive advantage in a timely manner (Gandossy, 2003; Sharifi & Zhang, 1999). The capabilities in agile organizations are highly dependent upon the learning process and ability of

employees carrying out operations and completing tasks in the shortest possible time (Sherehiy et al., 2007). With speed, agile organizations require using resources effectively, providing more opportunities to meet customer satisfaction (Chonko & Jones, 2005; Shapiro, 2001; Vesey, 1992).

Individuals in agile organizations swiftly develop new skills and abilities including quickly adapting to new development approaches and innovative management skills (Breu et al., 2002). For agile in-house development, the collective abilities to quickly learn, adapt and respond to innovations ahead of the competition are key developer-embodied factors.

Agile software development environments focus on the high talents and skills of individuals including ability to learn and swiftly develop high capabilities and expertise to help teams accomplish their tasks and projects (Cockburn & Highsmith, 2001).

According to Shinkle (2009), in agile software development, individuals go through various skill levels to become experts in agile environments: novice, advanced beginner, competent, proficient and expert levels. Ability to learn and carry out tasks efficiently in the product development environment means they are able to deliver new products in the shortest possible time (Sherehiy et al., 2007). According to Hopp and Van Oyen (2004), individual adaptability, speed improvement to do tasks, reduction in setup time (sprint-planning) and cross-functional training enabler shorter average development cycle times. Achieving shorter development cycles is dependent upon expertise built through learning in one's own agile software development (Shinkle, 2009).

Most important is the ability to learn and carry out tasks efficiently in a development environment delivering products in the shortest possible time (Sherehiy et al., 2007). Individual learning and flexibility, improvement in task speed, reduction in setup time and cross-training are the key factors facilitating shorter average development cycle times (Hopp & Van Oyen, 2004). The goal for shorter development cycles can only be achieved through expertise in agile software development (Shinkle, 2009).

Automation is a critical factor for short development cycles reducing development time, achieving development flexibility and delivering innovations (Gunasekaran, 1998). Upfront testing in agile software development driven by short development cycles requires automation (Shaye, 2008).

Agile software development requires high performing hardware and tools (Gunasekaran, 1999) to deliver features at a constant pace. Internet access for email,

collaboration systems, IM tools, net meeting and video conferencing, is critical resource for short development cycles allowing instant collaboration and feedback with industry experts, product managers and customers for rapid feedback (Phalnikar, Deshpande, & Joshi, 2009).

Hypothesis 5: Organizational Learning positively affects the Speed of agile software development teams or organizations.

3.5 Organizational Learning and Cooperative Teams (Team Effort)

Cooperative teams relate to collective effort (team effort) for decision making and undertaking various tasks (task sharing) rather than having individual or solo effort (Yauch, 2007). Hence, cooperative teams are based on empowerment, cross-functional teams and decentralised decision making (Yusuf et al., 1999).

Some of the major benefits of collective effort are mutual acceptance and understanding between different functional units on organizational goals and objectives, enhanced speed associated to carry out tasks and to make decisions, shared responsibility for team deliverables, loyalty and commitment and responsiveness capability (Sherehiy et al., 2007). The impact of collective effort is enormous in a dynamic product development environment as it helps to achieve product goals and objectives by carrying-out tasks based on collective thought and effort (Tjosvold & Tjosvold, 2015).

An agile organization, has an inbuilt approach to create, organize knowledge and learn around their daily work process and activities to continuously upskill their employees (Dodgson, 1993; Yeung, 1999). An agile organization encourages its employees to adopt the process of double-loop learning which allows employees to learn from one another on the job and produce better results (Pérez-Bustamante, 1999; Shrivastava, 1983).

The major benefits of organizational learning are (1) to become better and more responsive to the marketplace, (2) the parallel development effort while learning enables competitive advantage, (3) enhances competency organization-wide for tasks, and (4) enables having a workforce capable of adapting to changes (Appelbaum & Gallagher, 2000).

In agile software development, team effort involves two or more developers working together on a single task (task sharing). One of the major benefits identified is that developers learn more effectively in a team environment based on cooperation, especially when working with the entire team rather than working by themselves which often results in competing with others in the team (Brown & Palincsar, 1989).

Agile software development is based on team effort (cooperative teams) to carry-out tasks involved in identifying and developing software products. Hence, vision planning and roadmap planning (Sliger & Broderick, 2008; Wang, Hsu, Chen, & Lin, 2008; Werder et al., 2016) requires a planning or steering committee to make decisions collectively on proposed features for implementation (Lal, 2011).

Developing business cases can help a steering committee to make effective decisions by setting priorities, evaluating the business value of the software product and reducing the risks of developing software with the wrong requirements (Boehm, 2003). The involvement and cross-functional collaboration of the product management and software engineering units (including any other business function such as marketing and sales) with the steering committee (product planning team) ensures that the software project follows a critical path and as a result delivers the project successfully within budget and on time (Batra, Xia, VanderMeer, & Dutta, 2010).

The cooperative social process can be achieved through proper collaboration (example: developers working closely on a daily basis) and communication from team members who value and trust each other (Nerur et al., 2005). The communication and joint activities from two or more team members working together can reduce potential risks to the software project (Bardis, 1979). The entire team including the on-site customer must evaluate and explore decisions during product backlog meetings to determine what kind of software functionality needs to be added for each release and estimation to implement the functionality (Cobb, 2015).

The entire team evaluates each backlog item based on its risk, priority, value and necessity which requires a collaborative effort (Schwaber & Sutherland, 2012). The impact of having an on-site customer at an early stage can reduce the number of errors related to business requirements and improvement of product quality through better understanding about their needs (Mohammadi et al., 2008).

During sprint planning, the entire team with the on-site customer (product owner) selects a few user stories from the product backlog to be implemented in the sprint cycle.

Hence, the team collectively makes the commitment decision to deliver user stories (Cohn, 2004). The collaboration from sprint planning enables the entire team to organize, make commitments and perform the tasks that become shared team problems (Hansen & Baggesen, 2009). It requires a high level of trust to establish desired consistency, shared vision, commitment and teamwork between developers in agile teams (Cockburn & Highsmith, 2001).

In daily-scrum meetings, each developer has the opportunity to discuss and communicate about their work from the previous day, their objectives for the current day's work, and to identify issues that might prevent them completing their tasks (Maximini, 2015). After every team member reports their progress in the daily-scrum meeting, anyone from the team can propose a task that requires additional actions to be discussed further as part of the collaborative process (Stray, Moe, & Aurum, 2012).

The daily discussions with the team expand and evolve around the shared unsolved problems rather than arguing about the errors encountered in the past (Hansen & Baggesen, 2009). Because of the high involvement and participation, developers become motivated to perform the task with more commitment which may lead to innovation, more productivity and helping other developers (Fenton-O'Creevy, 1998; Moe et al., 2008).

The common practice from pair programming usually only involves two programmers to perform a common task. However, the collaboration of pair programming with an on-site customer and usability engineer may possibly lead to better understanding about the software project. The on-site customer should work very closely (possibly in the same room) with the developers in order to answer questions and provide instant feedback because the on-site customer possesses many of the critical knowledge about the project requirements (Koskela & Abrahamsson, 2004), while the usability engineer integrates user stories from the on-site customer with scenario-based design for unit testing, user acceptance testing and usability testing (Sohaib & Khan, 2010). The collaboration between developers, the on-site customer and the usability engineer can produce a usable product clearly identified from the user's perspective.

According to Melnik and Maurer (2004) a team effort involves face-to-face interaction facilitating a higher team velocity since they continuously support and learn from each other. At all levels (business and development) the experienced individuals of the team provide learning, knowledge and speed in agile development environments (Savolainen, Kuusela, & Vilavaara, 2010). According to Chen (2005), newcomers in the

agile team must be quick learners and adapt to the team culture. A study done by Hoegl and Proserpio (2004) shows close proximity enables higher levels of learning and meetings, a flow of rich information and better coordination of tasks and contributions.

Hypothesis 6: Organizational learning positively affects the Team Effort of agile software development teams or organizations.

3.6 Organizational Learning and Workforce Agility

Workforce agility acts as one of the main enablers for organizational agility, requiring the workforce to adapt to changes in the shortest possible time in order to deal with unpredictable changes at the marketplace (Qin & Nembhard, 2015; Sohrabi et al., 2014). A highly skilled and adaptable workforce will have the flexibility to be able to deal with non-routine tasks and problems (Youndt et al., 1996).

The attributes of workforce agility can be identified through employees'(1) positive attitudes towards continuous learning and self-development; (2) welcoming and accepting of new ideas and changes; (3) taking full responsibility for their work and cross-functional effort based upon empowerment; (4) effort and contribution in problem solving via knowledge management; (5) ability to work in different work environments; and (6) ability to take multiple roles (formal and informal) in a team setup (Dyer & Shafer, 2003; Griffin & Hesketh, 2003; Gunasekaran, 1999; Plonka, 1997; Sherehiy et al., 2007). In addition, workforce agility provides benefits such as enhancement in product quality, self-learning curve, customer service and identifying product scope accurately (Sherehiy et al., 2007).

Agile software development requires general skill set (generalists) individuals who are flexible and have multiple development and business skills to be able to work in agile development teams (McConnell, 2004). As such, individuals are required to be continuously learning through on-job training to develop new skills (Bottani, 2009).

According to Hopp and Van Oyen (2004), the generalist roles lead to workforce agility through high productivity, meeting frequent delivery commitments, reducing time for short development cycles, their understanding leading to better quality products with better features. Workforce agility often leads to a highly skilled and experienced development team (Cockburn, 2002).

In agile software development, development teams are cross-functional teams. The cross-functional team must have developers with at least the roles of systems analysts, designers and testers, the product owner and a scrum master (Lagerberg et al., 2013). Collaboration in cross-functional teams enables generalist skillsets through learning each other's jobs and roles (Marczak, Kwan, & Damian, 2009). The team members in cross-functional teams establish shared understanding to achieve effective communication and successful coordination to work collectively on tasks (Marczak et al., 2009).

Workforce agility leads development teams be adept at using multiple approaches to deliver software projects (Ashmore & Runyan, 2014). Workforce agility means that development teams and on site customers are required to collaborate and work together on a daily basis to accept and deal with unanticipated changes (Ashmore & Runyan, 2014).

According to Hopp and Van Oyen (2004), agile software development enforces cross-training and job sharing to enable individuals to learn to develop multi-skills to take other roles in their teams. Pair programming, mentoring and coaching are key practices for knowledge transfer to develop multi-skilled engineers (Goebel, 2009). Task sharing is critical in agile development environments to develop a general skill base.

Therefore, the successful implementation of workforce agility in agile software development can improve the overall process of software development and it can be achieved by using a common language and aims and consistent goals that are based on the innovation, ideas, knowledge, expertise and learning of diverse team members.

Hypothesis 7: Organizational Learning positively affects the Workforce agility of software development teams or organizations.

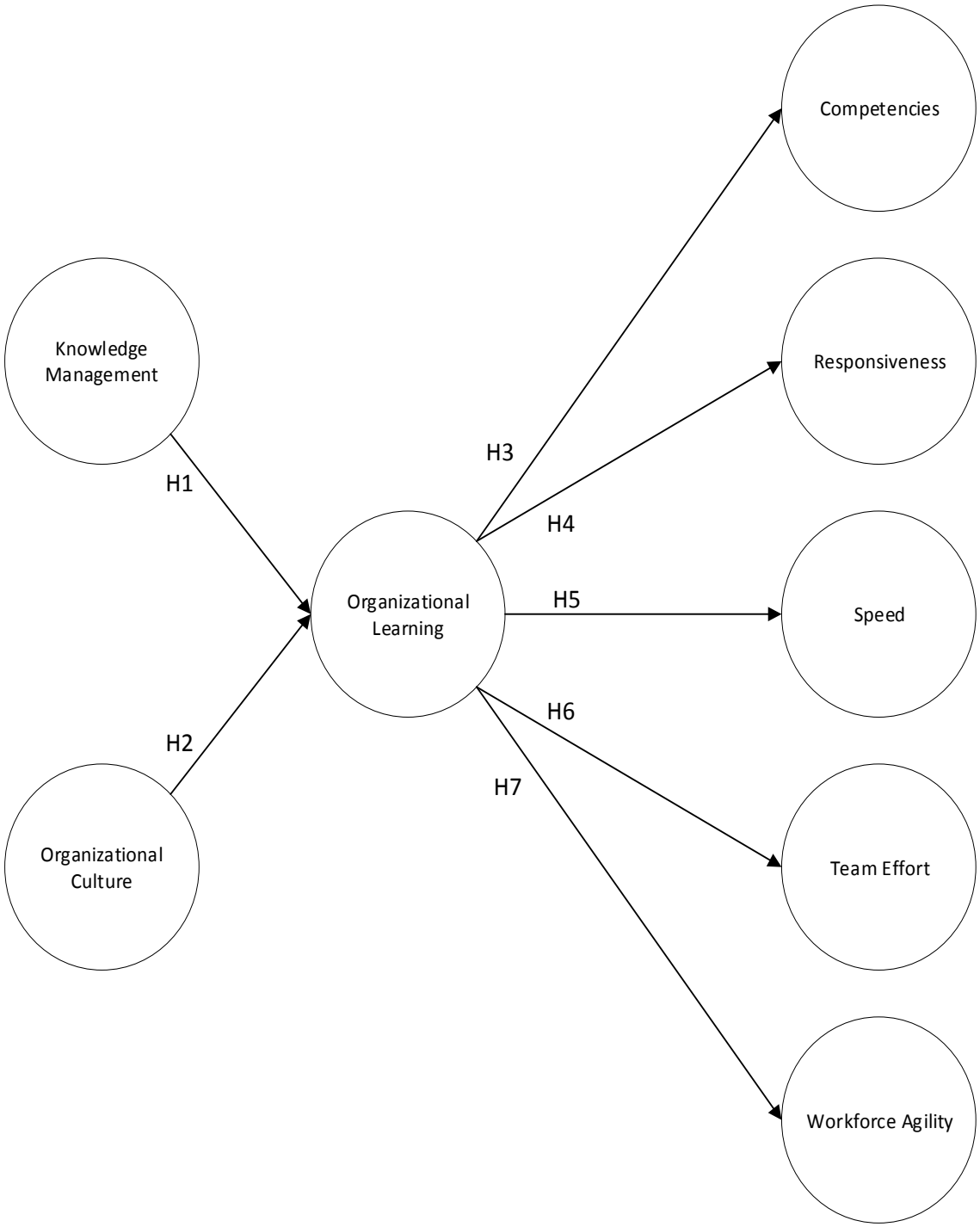


Figure 1. The research model and relationship of hypotheses

This chapter identified seven hypotheses as shown in Figure 1. The next chapter provides information about research methodology.

Chapter Four: Research Methodology

4.0 Chapter Overview

This chapter provides information on the research methodology, i.e. the overall research process, used for this study. First, information is provided on the research paradigm that was the basis for this study, followed by information on the research method used to answer the research questions and to achieve the research objective defined in Chapter 1. In addition, this chapter discusses the data collection method and data analysis technique used, including information on ethical considerations for this study.

4.1 Research Paradigm

The purpose of a research paradigm acts as guidance for the researchers to design and construct a model representing the study to ensure the study can be perceived by members of the research community (Johannesson & Perjons, 2014). A research paradigm is the belief that guides the researcher to conduct appropriate research, helping to establish a perspective from the community of researchers based on shared concepts, values, assumptions and practices (Johnson & Christensen, 2010). Hence, the research paradigm guides the researcher to formulate the research question(s), participant selection, data collection instruments and collection procedures, including the data analysis (Kivunja & Kuyini, 2017).

Generally, there are three main research paradigms: positivist, interpretive and critical theory approach (Creswell, 2009). Each research paradigm is characterized based on its philosophy to deal with the nature of truth and of reality (Ontology), knowledge and justification (Epistemology), and identification of a research method for the study (Methodology) (Johnson & Christensen, 2010).

The positivist paradigm assumes that the tested knowledge is derived by investigating the phenomenon, and the outcomes are predicted based on descriptive and factual statements (Scotland, 2012). The researcher observes the object as an independent entity, aiming to find meaning behind the discovered reality (Cohen, Manion, & Morrison, 2013; Crotty, 1998; Pring, 2000). The empirical investigation driven by the positivist approach uses a deductive approach (testing a theory) to generalize and make predictions (Scotland, 2012). Hence, the positivist approach may include a process to generate hypotheses, form correlation on models, quantify measures on controlled dependent and independent variables, and generalize from the sample (Chen &

Hirschheim, 2004). Following this assumption, the researcher can predict the outcome from natural human activities by collecting data from potential participants while making sure that there is no bias in the research approach (selection of participants, results interpretation etc.) (Bryman, 2012).

The key process of the interpretive approach is to construct and refine information from multiple assessments before integrating to create a new knowledge (Sarantakos, 2005). Unlike the positivist approach, the interpretive approach does not require defining of dependent and independent variables before conducting the research but focuses on the opinions and perceptions of people based on their observations and interviews (Kaplan & Maxwell, 2005). The aim of interpretive methodology is to have an understanding of individuals' perspectives including interaction between individuals and their behaviour (Creswell, 2009). Therefore, interpretive researchers require to interpret information through direct social interaction with people's consciousness, language and shared meanings (Myers, 2013).

In contrast with the other paradigms, the critical theory paradigm assumes that reality is derived from a combination of social, political, cultural, economic, ethical and gender values (Guba & Lincoln, 1994). The main purpose of the critical theory approach is to examine the values and assumptions of the participants when engaging in a social action (Crotty, 1998; Scotland, 2012). The critical theory method generates data similar to the interpretive paradigm, however explicit values are now placed on the thematic interpretation of data (Scotland, 2012).

For this study, positivist approach is identified to be the most suitable among the three paradigms described above to investigate the research problem. Following the positivist paradigm, this study attempts to investigate software development practices that enable agility for the entire organization. Hence, a list of agile method practices will be identified which ought to be considered to improve or change their development environment to successfully develop and deliver software products.

4.2 Research Method

Generally, there are three types of research methods: qualitative, quantitative and mixed method. The main difference between qualitative and quantitative study is that one focuses on closed-ended questions and relies on numerical data in order to conduct statistical analysis and interpret the result, while the other focuses on open-ended questions where the researcher collects the data during the engagement with participants

directly (Creswell, 2013). The mixed method is an approach that uses both qualitative and quantitative methods. Most researchers tend to use the mixed method because it helps them to provide a complete understanding of their research problem through diverse types of data being collected, rather than use quantitative or qualitative data alone (Creswell, 2013).

The purpose of a research helps the researcher to determine the kind of knowledge to be produced using a specific research method (Blaikie, 2009). The purpose of a research can be differentiated into two types: exploratory and explanatory. Exploratory research is used when there is little information available on the research problem so that the researcher can provide clear understanding of the research problem (Blaikie, 2009). Explanatory research provides rich insights and seek patterns on observed phenomena, behaviour, relationships, attitudes, social processes and social structures (Blaikie, 2009).

This is an exploratory study using the survey (quantitative) method. Hence, this study uses the widely available literature on agile organizations and agile software development methods to investigate and identify the agile software development practices that lead to agility based on an agile organization's concepts. Hence, this study identifies agile software development practices that software organizations and vendors 'ought to consider' for their development environment to successfully develop and deliver software products and to become market leaders.

This study chooses the quantitative method to answer the research questions and to examine the relationship between the hypotheses. From the perspective of a quantitative study, it provides guidance for the researcher to collect data and develop numerical measurements from observations in order to produce generalizations from statistical evidence (Creswell, 2013).

4.3 Data Collection Methods

The survey method is used for this study because it enables the researcher to involve widely targeted participants for this study. The survey method also ensures that there is no data manipulation from participants during the data collection process which is one of the characteristics of a survey (Creswell, 2013). While in the experimental study, the researcher attempts to control all variables that may affect the result: however it is still one of the most important approaches to explain causality from the manipulation of an independent variable to expect the desired result on the dependent variable (Singh, 2007).

Web-Based Survey

There are several reasons for using a web-based survey for this investigation compared to other survey approaches such as a paper-based survey or an email-based survey. First, the participants identified in this study are software development organizations and development teams and most of them were working during work days. Consequently, the researcher used a web-based survey to provide easy access for these participants to be part of this research by using the internet whenever and from wherever it suited the participants, and also it can be useful for those who may face difficulty participating through other channels (Wellman, 1997).

Second, a web-based survey creates distance between the researcher and the participants. Therefore, there is no interference from the researcher when participants are filling out the questionnaire. If the participants know the researcher or vice versa, the responses from a web-based survey may become biased and this could affect the results of the study (Andrews, Nonnecke, & Preece, 2007).

Third, the web-based survey decreases the time and cost spent on distributing the questionnaire to a large number of participants (Garton, Haythornthwaite, & Wellman, 1997; Yun & Trumbo, 2006). The researcher can reach participants separated by geographic distance in a short time using a web-based survey while it would consume more time to collect the equivalent amount of data in a face-to-face study environment (Wright, 2005). The web-based survey eliminates the costs of paper and other expenses such as travel and recording tools (Wright, 2005).

Fourth, a web-based survey enables the researcher to ensure quality responses from the participants by designing the questionnaire using a systematic approach and having flexibility with the design. For example, participants are required to answer a question on a behaviour, which will also have options that the participant may use if the question does not fit with their situation. As a result, it helps to prevent or reduce the risks of missing data (Andrews et al., 2007). In addition, there are settings in a web-based survey to prevent the submission of multiple responses from same IP address or required e-mail address.

Lastly, data analysis becomes easier because the participants' responses from a web-based survey are usually represented in a database automatically which eliminates the risks of transcription errors. In addition, the researchers can export the data to a desired format and use it for data analysis.

Sampling

Sampling is the process of selecting representatives from a population and usually the researcher wants to generalize the characteristics of a subset from a large group so they can make statements about the population based on the study's sample (Johnson & Christensen, 2010). The purpose of sampling is to enable researchers to attain their research goal since it is impossible to study every single member in a population because of cost and time to collect the data (Muijs, 2010).

There are two main types of sampling methods: probability sampling and non-probability sampling. Probability sampling refers to a selection procedure of members from the population having a chance to be selected and the result can be used to reflect on the entire population (Reis & Judd, 2000). Non-probability sampling refers to a selection procedure where participants are not randomly selected from the population, often used by researchers in special cases (Reis & Judd, 2000). As discussed in the previous chapter, the sampling frame of this study was only agile software organizations or development teams and only those that were using agile software development approaches or methods. especially in New Zealand and Australia.

The companies were identified through an index website that contains the software company name and contact information in New Zealand and Australia. Each of the agile software development teams or organizations were invited through an email. Some of the potential participants' information was also acquired from contact information on NZ agile conference members.

Respondents were identified through agile community groups and various agile software development conferences held in New Zealand and Australia. The following are the URL address of some of them:

1. New Zealand agile conference (http://agilenz.co.nz/)
2. Agile Australia conference (http://www.agileaustralia.com.au/2016/)
3. Agile Auckland (http://www.meetup.com/Agile-Auckland/)
4. Agile Wellington(http://www.meetup.com/AgileWelly/)
5. Agile Brisbane (http://www.meetup.com/Agile-Brisbane/)

4.4 Ethical Considerations

Ethical principles in research methodology are important, reflecting the researcher's professional behaviour when conducting a research. The purpose of ethical considerations is to protect all parties involved in the data collection and the information used from the findings of study, especially the researcher, the participants and the researcher's institution (Creswell, 2009).

This study was undertaken following approval from AUTEC (Auckland University of Technology Ethics Committee). The ethics approval was required before conducting the web-based survey and information about the research was provided on the information sheet for the participants. The information sheet is available in appendix A

From an ethical perspective, this study informed all participants that their participation was voluntary. Before taking the questionnaire, the participants were informed that their responses from the questionnaire were completely anonymous and the information provided by them would only be used for this study. During the process of answering the questionnaire, participants had the right to withdraw their participation at any time if they felt uncomfortable. In addition, participants' confidentiality and privacy were protected and data related to this study was kept in a secure place which could only be accessed by the researchers involved in this study.

4.5 Development of the Survey

Initially, the survey questions were developed based on a literature review and aimed to address the 7 hypotheses and consisted of 120 questions. After revision and improvement with the supervisor, 76 questions were selected and divided into nine sections (reflecting themes of agile organization concepts). Then, a pre-test survey (pilot study) was conducted with the involvement of one master's student with similar studies and two lecturers who are experts in agile software development. According to Fowler (2009), the purpose of pre-testing a survey is to discover how reliable the survey instruments and data collection protocols work under realistic conditions. Additionally, the pre-test of the survey questionnaire ensured the questions were clear and were likely to obtain the desired information (Gaddis, 1998). The questionnaire was then sent to AUTEC for approval before sending the email invitation to the participants.

The questionnaire was structured in the following sequence:

At the beginning of the questionnaire, there was a welcome page that provided brief information about this study followed by an information sheet which was also approved by AUTEC. The detail of the information sheet is available in Appendix A. Based on the pre-test survey, it was estimated that it would take around 20 to 25 minutes for participants to complete the survey. The full version of the survey questionnaire is provided in Appendix B.

Table 8 below explains the agile software development practices with reflected questions in the survey.

Table 8. Agile Software Development Practices with Reflected Questions

Agile organization concepts	Agile software development practices	Reflected in Survey
1. Cooperative Teams	• Vision planning • Product backlog • Daily stand-up meeting • Pair programming • Sprint planning • Coding standard • Retrospective	Question 12-33
2. Competencies	• Continuous integration • Self-organizing teams • Product backlog • Test-driven development • User stories • Unit testing • User story acceptance testing	Question 34-45
3. Responsiveness	• Product backlog • Release planning • Sprint planning • Retrospective • User story acceptance testing	Question 46-49
4. Organizational Learning	• Vision planning • Retrospective • Release planning • Sprint planning • Retrospective • Test-driven development	Question 50-56
5. Organizational Culture	• Open workspace • Coding standard • Collective code ownership • Retrospective • User story acceptance testing	Question 57-62
6. Workforce Agility	• Self-organizing teams • Daily stand-up meeting • Sprint planning • Retrospective	Question 63-65

	• User stories	
7. Speed	• Sprint planning • Product backlog • Test-driven development • Continuous integration • Refactoring	Question 66-72
8. Knowledge Management	• Pair programming • Test-driven development • Retrospective • User story acceptance testing • Regression testing • Refactoring • Coding standard	Question 73-76

4.6 Survey Administration

An invitation message regarding the recruitment of participants for the survey was sent through email. The invitation message indicated that this study was part of a master's thesis and the objective of this study was to investigate the agility elements from the software development industry and identify factors (specific software development practices required to achieve agility) that software organizations and vendors ought to consider improving or changing in their development environment. It also informed participants that the survey was anonymous and voluntary. In addition, their identities would not be included for any other study.

The survey was made available online on Google Survey between May 2017 and July 2017. The survey was closed on 10 July 2017. There were 63 responses gathered in total. The collected data is analysed in detail in Chapter Five.

4.7 Data Analysis Techniques

After collecting the data from participants, the dataset was analysed using Structural Equation Modelling (SEM). SEM is used by IS researchers to analyse data to meet the standard criteria with utilizing high quality statistical analysis from this second generation data analysis technique (Gefen, Straub, & Boudreau, 2000). The difference between SEM and first generation statistical tools such as regression analysis is that at first researchers need to build a relationship model between dependent and independent variables in order to obtain answers from a collection of research questions (Anderson & Gerbing, 1988). Then the analysis result displayed to the researcher is presented in simple form, systematic, including comprehensive analysis to test the validity of the SEM model (Gefen et al., 2000). There are two parts that the researcher has to include in the SEM

model: structural model and measurement model (Gefen et al., 2000; Mehlsen, 2009). The structural model explains the correlation among the latent variables as this connection is the main purpose of the analysis while the measurement model describes the connection between the latent variables with manifest indicators (Mehlsen, 2009). The reason why SEM requires a measurement model is to assist latent variables because they cannot be measured directly, so manifest indicators serve as measurement variables in the questionnaire (Mehlsen, 2009). The analysis from a combination of structural and measurement models allows the researcher to analyse measurement errors on the observed variables which acts as an integral part of the model and also evaluates the hypothesis testing from variable relationships (Gefen et al., 2000).

4.7.1 Structural Equation Modelling

Generally, there are two types of SEM methodologies that can produce formative measurement models with latent variables: Covariance-Based SEM (CB-SEM) which can be analysed by using the software packages of AMOS and LISREL; and Partial Least Squared (PLS) which used a PLS-Graph and SmartPLS (Wong, 2013a). This study selected PLS over CB-SEM for two reasons. First, it is more appropriate to use PLS when the research model is in an early stage. Second, this research model has formative constructs and PLS can estimate formative constructs (Chin & Newsted, 1999).

As discussed in section 4.2, this study chose exploratory as the research objective and it is really suitable with the role of PLS to assist the researcher on exploratory research whose main goal is to predict or identify the latent variable while CB-SEM is more suitable when it is applied in theory testing, theory confirmation and other forms of comparison (Hair et al., 2011; Wilburn, 1984; Wong, 2013a).

The advantage of CB-SEM is that it can reduce the effect of measurement errors and also separate each error from other sources which makes it easier for the researcher to analyze the data than with PLS (Kotzab, Seuring, Müller, & Reiner, 2006). PLS provides a less accurate estimation of errors in measurement models compared to CB-SEM which provides higher quality criteria for the overall measurement model (Gänswein, 2011).

To sum up, this study used PLS as the most suitable data analytical technique. PLS uses SmartPLS because it has the capability to work with SPSS and has a user-friendly graphical interface.

Table 9. Comparison Between the Approaches of PLS-SEM and CB-SEM

(Chin & Newsted, 1999; Codita, 2011)

Criteria	**PLS**	**Covariance-based SEM**
Fundamental method	Variance-based	Covariance-based
Objective	Prediction oriented	Parameter oriented
Data distribution assumptions	Predictor specification (non-parametric)	Multivariate normal distribution and independent observation (parametric)
Model evaluation	Heuristic method	Statistical "fit" measures
Relationship between the indicators and the construct	Formative and reflective	Typically reflective
Parameter estimates	Consistent as indicators and sample size increase (consistency at large)	Consistent
Interdependence between constructs	Not possible in the basic model	Possible
Sample Size	Small sizes are admissible under appropriate sample power considerations	Depending on the complexity of the model, large sizes are mandatory
Implications	Optimal for prediction accuracy	Optimal for parameter accuracy

Data Screening

Data Screening refers to the initial process of data examination to ensure the quality of data and data accuracy (Tavakoli, 2012). The purpose of data screening is to find any error or inconsistency in the data that may affect the research findings (reference). The data from participants is considered as invalid and is not included in the data analysis when the participants leave a question unanswered in the questionnaire or submit an incomplete questionnaire. The process of data screening can be considered as time consuming when the researchers who conduct data collection and data entry are different researchers (Goyal, 2010). However, with this research the primary researcher was the one who was responsible for the data collection and entry stages for this study. In addition, data screening helps the researcher to comprehend the fundamental relationships

between variables and ensures the responses from intended participants meet all requirements and are valid before the data is transferred into the data analysis stage.

Outliers (Extreme observation)

Outliers is one of the processes included in the data screening to identify and remove or replace critical data that are different from others (Jegerski & VanPatten, 2013). There are three main steps to handle outliers: find the outliers from data entry, determine the cause of outliers and reduce the impact of outliers (Tinsley & Brown, 2000). The purpose of identifying outliers is to ensure that the data are reflected on the normal distribution within the population because if not, it may affect the statistical results, especially in quantitative research (Loewen & Plonsky, 2015). There are two ways to reduce the impact of outliers: univariate (case with extreme value on single variable) or multivariate outliers (unusual case between combination of scores on two or more variables) (Tinsley & Brown, 2000).

Multivariate Analysis

This study chose normality to ensure that the shape of the data distribution was related to a normal distribution before examining the sample data further in the PLS. Following the outlier, normality analysis also includes the same terms: univariate and multivariate normality. The evaluation of univariate normality is already sufficient during data screening and is used in most cases when the distribution value is symmetrical while the evaluation of multivariate normality is used when each variable in the data model is normally distributed with a fixed value (Nimon, 2012).

If the data is considered multivariate normality, it means the variables are also valid in univariate normality (Stamatis, 2002). However, the evaluation of multivariate normality itself is difficult to achieve and only necessary when it is critical (Stamatis, 2002). Therefore, this study used univariate normality as the initial step of multivariate analysis then proceeded to multivariate normality when it was required. There are two methods that the researcher can use to examine univariate normality which involve graphical tests (histograms and normality probability plots) and non-graphical tests (Shapiro-Wilks test, the evaluation of kurtosis and skewness values) (Nimon, 2012).

4.7.2 Specification of Measurement Model in SEM

The measurement model in SEM indicates that the relationships between each latent variable and indicator variables have to be specified logically and systematically for each construct in the structural model (Döscher, 2014). There are two approaches to construct a measurement model in SEM: formative and reflective measurement models. Determining the measurement model between formative and reflective is very crucial because it serves as one of the requirements to establish the relationship between variables in the structural model or it may end up with the researcher producing a misleading result from the incorrect interpretation of the research theory (Coltman, Devinney, Midgley, & Venaik, 2008).

This study chose the formative measurement model as the most suitable approach to construct the measurement model in SEM. There are several theoretical and empirical considerations which specify whether the construct of the measurement model is formative or reflective (Coltman et al., 2008).

The first priority of the theoretical consideration to identify the measurement model is the direction of causality between items and latent constructs. In the reflective model, the causality is formed from the construct to the items which indicates that one construct can cause many variations in the item measures or show that from the existing latent construct, it can produce an independent measurement (Bollen & Lennox, 1991). The formative model has the opposite effect where a variation in item measure causes a variation in the construct or combination of indicator variables causing the latent construct (Jarvis, MacKenzie, & Podsakoff, 2003).

Still following the theoretical consideration, the next priority is the characteristics of used items to measure the construct. The reflective model can be identified from the interchangeable used items where adding or removing an item does not affect the conceptual domain of the construct and also shares a common attribute (Coltman et al., 2008). On the other hand, the formative model can be observed through a unique attribute for each item and the items are not interchangeable so if there are some changes, they could affect the conceptual domain of the construct (Jarvis et al., 2003).

From the empirical consideration, each item in the reflective model should be interrelated with a high positive value and it can be tested using internal consistency and reliability from Cronbach alpha (Coltman et al., 2008), while in the formative model, the interrelation can contain any value with the minimum provision of having the same

directional relationship. In addition, the reflective model can identify the measurement error in the construct by using confirmatory factor analysis to find the error while the formative model cannot identify the measurement error. However, vanishing tetrad tests are recommended to predict the formative items as intended (Coltman et al., 2008).

The next priority coming from empirical considerations is that items in the reflective model have identical signs of relationships with consequences or antecedents as the construct while it may appear not identical for the formative model (Jarvis et al., 2003). In order to achieve this consideration, the researcher can test the content validity for the reflective model or nomological validity for the formative model (Coltman et al., 2008).

4.7.3 The Main Processes of PLS

There are three sets of relationships that can be found in the latent variable path models in PLS: inner model, outer model and weight relations (Chin & Newsted, 1999). The inner model describes the relationship between the latent variables based on the theory, while the outer model defines the relationship of each indicator that relates to its latent variable. Weight relations need to be defined in order to complete the specification for the PLS estimation algorithm based on these two models.

There are three weighting schemes that are available to use in PLS: centroid scheme, factorial scheme and structural or path weighting scheme. This study chose the path weighting scheme because it was considered the best in terms of describing models and also provides the highest R^2 value for endogenous latent variables which are applicable for all kinds of path model specifications and estimations (Hair, Hult, Ringle, & Sarstedt, 2016; Vinzi, Chin, Henseler, & Wang, 2010).

Validating the measurement model

The measurement model attempts to represent the connection between observed variables and latent variables. The objective of testing the measurement model is to state how the latent variables are measured, which is based on observed variables and used to define the measurement properties of the observed variables (validity and reliability).

Reliability Considerations

Reliability ensures that the portion of items used to measure a construct is related to be considered as a set of items. The reliability of a construct is calculated separately and assessed independently which is based on internal consistency (Netemeyer, Bearden, &

Sharma, 2003). This study used composite reliability (CR) to assess the internal consistency from a set of items in PLS. The results can vary between 0 and 1. The recommended value of CR for each construct should be greater than 0.70 in order to reach a satisfactory level and a higher value of CR indicates higher reliability (Hair et al., 2011).

Discriminant and Convergent Validities

Discriminant validity indicates that each of the constructs is distinct from the other. The result can be assessed by using the recommendation provided by Gefen, Straub, and Rigdon (2011), which compares the square root of the average variance extracted (AVE) values for each construct which should be greater than the corresponding correlations between this construct and any other constructs.

Convergent validity measures the correlation between a construct's indicators and the alternative indicators of the same construct. It can be assessed by using AVE which acts as the summary indicator among a set of items measuring the same construct. The recommended value of AVE should be greater than 0.50 in order to reach a satisfactory level (Wong, 2013b).

Validating the structural model

The structural model measures the validity of the measurement model and tests the relationships among theoretical latent variables (Schumacker & Lomax, 2010). The structural model and hypotheses are evaluated through the coefficient of determination (R^2) of the dependent variable, which measures its variance based on the independent variables in the structural model. The recommended value for R^2 should be above 0.2 which indicates that the structural model is valid and independent variables successfully predict the changes of dependent variables (Wong, 2013b).

PLS does not evaluate the goodness of fit metric for the measurement and structural model. The purpose of model fitting is to emphasize the overall model fit with the entire indicator variables with hypothesized covariance from the measurement model to form the best possible construct and the researcher can use various statistical tests from the fit indices in order to determine if the model fits the data (Gefen et al., 2000).

According to Chin (2000), there are two reasons for considering model fit in PLS analysis. Firstly, the goodness of fit measurement is related to the model and requires all measures as reflective, however the construct model used in this study is being modelled as formative. Secondly, it does not predict and relate to how well the parameter estimates

of latent variables or any item measures. As a result, this study does not include indices during PLS analysis to avoid confusion.

This chapter provides information on the research methodology used for this study. This study was driven by the positivist approach. For the data collection, this study used a web-based survey targeting participants from the software development organizations and development teams. The data analysis was done using the PLS method using SmartPLS as the software tool.

Chapter Five: Data Analysis and Findings

5.0 Survey Results

This section provides results from the survey, and includes statistical analysis done using SmartPLS.

Section 5.1 – 5.11 provide general results of the survey on agile method practices and section 5.12 – 5.14 provide results on the hypotheses tested through this survey investigation.

5.1 Targeted Participants

Table 10 provides data on the actual participants who were invited to take part in this survey, including data on those who took part in the survey. The result shows that there was almost a 35% success rate in terms of actual participation for this study. There were 181 invitations emailed out to the prospective participants (in-house agile software development teams, agile software development contracting companies and agile software vendors), mostly in New Zealand and some in Australia. However, only 63 participants responded to the online survey. From the 63 responses, 58 were screened to be suitable for data analysis. With 58 responses from different agile software development organizations, there were a sufficient number of responses for data analysis to able to identify agile organizational practices for gaining agility in software development.

Table 10. Survey Information

Survey Information	Responses	Percentage
Survey Invitations Sent	181	100.0%
Submitted Responses	63	34.8%
Valid Responses	58	32.0%

Participant Break-down

Table 11 shows that almost 33% of respondents were from in-house software development teams of business organizations and institutions. In addition, 43% of the respondents were software vendor organizations and 24% were from contracting software development organizations. Hence, this survey involved participants from organizations

that are using one of three ways the software is most likely to be developed and made available for business use.

Table 11. Organization Category

Organization Category	Responses	Percentage
In-house development teams	19	32.8%
Software vendors	25	43.1%
Contracting IT/Software development teams	14	24.1%

5.2 Team Size

Table 12 shows that almost 52% of the respondents had agile software development teams with 10 to 50 individuals. In addition, 24% of the respondents indicated that they had more than 50 individuals in their agile software development teams. The other 24% had less than 10 individuals in their agile software development teams. Hence, this survey involves participants from both large and small agile software development teams and organizations, so the findings will be applicable and can be used by all agile software development practitioners to improve their software development environment for agility and as a result achieve the benefits at the market place.

Table 12. Team Size

Team Size	Responses	Percentage
Less than 10	14	24.1%
10 – 50	30	51.7%
Over 50	14	24.1%

Team Size Break-down

Table 13 shows that for the in-house software development teams, an overwhelming 89% of respondents were part of large agile software development teams (i.e. 47% of respondents indicated that they were part of agile software development teams which had 10 to 50 team members and 32% indicated that they were part of agile software development teams that had over 50 team members). Furthermore, for software vendors 80% (68% + 12%) of the respondents were part of large agile software development teams that had from 10 to 50 team members). For contracting development teams, 64% of the respondents were part of large agile software development teams. On the other hand, almost 36% of respondents were from agile software development teams that had less

than 10 team members. The large number of respondents from large team sizes shows that agile software development is now regarded as a useful development approach for providing an effective work breakdown structure not only for small size teams but also for large size teams. When agile software development emerged in the 1990s, it was seen as a useful approach for small sized teams.

Table 13. Team Size Break-down

Team Size Break-down	Responses	Percentage
For In-house development teams		
Less than 10	4	21.1%
10 – 50	9	47.4%
Over 50	6	31.6%
For Software vendors		
Less than 10	5	20.0%
10 – 50	17	68.0%
Over 50	3	12.0%
For Contracting development teams		
Less than 10	5	35.7%
10 – 50	4	28.6%
Over 50	5	35.7%

5.3 Experience in Agile Environment

Table 14 provides statistics on the agile software development experience of the respondents. As shown in the table 14, almost 86% of the respondents had more than two years of agile software development experience. Hence, with a high number of the survey respondents with a good level of agile software development experience it is expected that it would lead to significant survey results, which will have both practical and theoretical benefits.

Table 14. Experience in Agile Environment

Experience in Agile Environment	Responses	Percentage
1-2 years	8	13.8%
2-3 years	9	15.5%
3-5 years	20	34.5%
5-10 years	15	25.9%
10-15 years	4	6.9%
>15 years	2	3.4%

Table 15 provides a further breakdown of the respondents on agile software development experience for in-house software development teams, software vendors and contact development teams. Most of the respondents for each of these different options for software development had more than two years of software development experience (for in-house development teams almost 90%, software vendors 88% and contracting development teams almost 79%). Having a significantly large number of respondents with more than two years of agile software development experience is critical to ensure the external validity of the survey results.

Table 15. Experience in Agile Environment Break-down

Experience in Agile Environment Break-down	Responses	Percentage
For In-house development teams		
1-2 years	2	10.5%
2-3 years	2	10.5%
3-5 years	8	42.1%
5-10 years	5	26.4%
10-15 years	2	10.5%
For Software vendors		
1-2 years	3	12.0%
2-3 years	3	12.0%
3-5 years	11	44.0%
5-10 years	6	24.0%
10-15 years	1	4.0%
>15 years	1	4.0%
For Contracting development teams		
1-2 years	3	21.4%
2-3 years	4	28.6%
3-5 years	1	7.1%
5-10 years	4	28.6%
10-15 years	1	7.1%
>15 years	1	7.1%

5.4 Projects Undertaken in a Year

Table 16 shows that 43% of respondents indicated that they had undertaken between four and six agile software development projects in a year and another 43% of the respondents had undertaken more than six agile software development projects. Just 14% of the respondents indicated that they undertook between 1 and 3 agile projects per year. Thus, most respondents would have significant understanding and skills on applying agile software development approach/practices in software development projects. In addition, the data presented in Table 17 shows that the agile software development projects are mostly short duration in nature based on the number of projects they undertook in a year, including the survey respondents had a solid experience and understanding in undertaking agile software development projects. It is not a surprising to see software vendors taking more than six projects in a year, since producing and selling software is their business and they have to keep ahead of competitors.

Table 16. Projects Undertaken in a Year

Projects Undertaken in a Year	Responses	Percentage
1-3 projects	8	13.8%
4-6 projects	25	43.1%
>6 projects	25	43.1%

Table 17. Projects Undertaken in a Year Break-down

Projects Undertaken in a Year Break-down	Responses	Percentage
For In-house development teams		
1-3 projects	2	10.5%
4-6 projects	9	47.4%
>6 projects	8	42.1%
For Software vendors		
1-3 projects	1	4.0%
4-6 projects	11	44.0%
>6 projects	13	52.0%
For Contracting development teams		
1-3 projects	5	35.7%
4-6 projects	5	35.7%
>6 projects	4	28.6%

5.5 Success Rate of the Projects

Table 18 shows that almost 90% of respondents had over 75% success with agile software development projects. Interestingly almost 50% of the respondents indicated that they have more than 90% success rate with their agile software development projects. This shows that the agile approach for software development is making a difference to the ability for the development teams to deliver projects successfully. It is well known that prior to the emergence of the agile approach, the success rate with software development projects was around 30%.

Table 18. Success Rate of the Projects

Success Rate of the projects	Responses	Percentage
<30%	1	1.7%
30%-50%	0	0.0%
51%-75%	4	6.9%
76%-90%	25	43.1%
91%-99%	27	46.6%
100%	1	1.7%

Table 19 shows that software vendors and contract development teams had 100% success rate with their agile software development projects while in-house development teams were not far behind with almost 94% success rate. These success rates clearly suggest that the agile approach is heading towards becoming a mainstream software development approach. This shows the significance of not only this research but any other research into agile software development, providing valuable information for successful agile adoption.

Table 19. Success Rate of the Projects Break-down

Success Rate of the Projects Break-down	Responses	Percentage
For In-house development teams		
<30%	1	5.3%
51%-75%	1	5.3%
76%-90%	9	47.4%
91%-99%	8	42.1%
For Software vendors		
51%-75%	1	4.0%
76%-90%	10	40.0%

91%-99%	14	56.0%
For Contracting development teams		
51%-75%	2	14.3%
76%-90%	6	42.9%
91%-99%	5	35.7%
100%	1	7.1%

5.6 Agile Practice Used

Table 20 provides data on some of the key agile method practices which ought to bring development success. Data on product backlog (almost 90%), sprint backlog (86%) and daily stand-up meetings (86%) shows that planning in agile software development is critical compared with other software development approaches; hence, successful agile software development must be driven by planning. The release planning (stated by almost 73% of respondents to be part of their development environment) practices may not rate as a high value practice for in-house software development teams compared to a software vendor environment or even with contract development teams since anything that is produced through in-house software development is usually for internal use only.

In a software vendor environment, a release plan is based on the roadmap plan for a time period capturing the strategic dates for market release up to two years. Similar Table 20 shows that the vision planning is ranked 10th overall in the table in terms of the practices adoption with only 60.30% of respondents indicating its adoption in their development environment. This is critical practice in a software vendor environment since they need to ensure any new development is well thought out, needed and will sell well at the marketplace.

It is worthwhile to note a high level of adoption (almost 73% of respondents stated that it is part of their development environment) rate for pair programming practices in agile software development projects. Often this practice has been misunderstood due to the cost assumption when two developers are working collectively on a single task. However, this practice ought to lead to a shorter implementation time, quality features, learning from each other when task sharing and having quality assurance work done upfront with implementation.

Agile method practices such as continuous integration (69%), retrospective (69%) and refactoring (65.5%) ought to be amongst the critical agile software development

practices but the level of data on adoption does not suggest every agile approach adopter sees these three practices in same light for their development environment. Continuous integration practice through automation is vital for short development cycles in projects whereas refactoring practice enables target performance requirements of the software product. In addition, retrospective practice is vital for reflection to get better from one short development cycle to the next including from one project to another.

It is interesting to note that with the test-driven development (TDD) practices, only 41.40% of respondents indicated that they had it as part of their development mind set. One possible explanation for this result could be that the quality assurance and testing tasks are still treated as separate activities from sprints. Another possible explanation is that most development environments still require significant levels of manual testing and quality assurance tasks. However, there appears to be a good level of adoption for TDD-related practices such as acceptance testing (62.1%), unit testing (almost 57%) and regression testing (55.2%).

These testing practices ought to be automated in an agile software development environment and are key for continuous delivery of working codes in short development cycles to be deployed in a production environment. Low levels of adoption for coding standards (almost 40%) and collective code ownership (almost 28%) show that agile software development requires to be discipline driven. As a result, the software development community is being advocated with the prominence of scaled agile methods especially with the emerging of cloud technology to host and offer the software product to a vast audience worldwide.

Table 20. Agile Method Practices

Agile Method Practices	Responses	Percentage
Product backlog	52	89.7%
Sprint backlog	50	86.2%
Daily stand-up meeting	50	86.2%
Release planning	42	72.4%
Pair programming	42	72.4%
Continuous integration	40	69.0%
Retrospective	40	69.0%
Refactoring	38	65.5%
Acceptance testing	36	62.1%
Vision planning	35	60.3%

Unit testing	33	56.9%
Regression testing	32	55.2%
Common workspace	24	41.4%
Test-driven development	24	41.4%
Coding standard	23	39.7%
User story mapping	20	34.5%
Self-organizing team	18	31.0%
Collective code ownership	16	27.6%
Burn-down/Burn-up chart	13	22.4%

5.7 Single Agile Method

Table 21 shows that almost 52% of respondents indicated that they had adopted a single agile method. The SCRUM method is the most popular method, which is not surprising since it provides solid project planning practices (product backlog and sprint planning) which create visibility organization-wide of the total amount of work that would be carried out including making it easier to execute, monitor and control the work in short development cycles. The next most adopted single agile methods are in fact two methods: XP (20%) and Kanban (20%). They both are strong with practices on development and delivery of projects.

Table 21. Single Agile Method

Single Agile Method	**Responses**	**Percentage**
Extreme programming (XP)	6	20.0%
Scrum	13	43.3%
Adaptive software development	3	10.0%
Feature-driven development	1	3.3%
Kanban	6	20.0%
Agile modelling	1	3.3%
Total Responses	**30**	**51.7%**

Table 22 provides data on single method adoption in the three different types (in-house development teams, software vendors and contract development teams) of development environments. The high adoption rate of the SCRUM method in all three different environments suggests that project planning is considered vital for development success regardless of the type of the development environment.

Table 22. Single Agile Method Break-down

Single Agile Method Break-down	Responses	Percentage
For In-house development teams		
Extreme programming (XP)	2	25.0%
Scrum	3	37.5%
Feature-Driven Development (FDD)	1	12.5%
Kanban	1	12.5%
Agile modelling	1	12.5%
For Software vendors		
Extreme programming (XP)	1	10.0%
Scrum	5	50.0%
Adaptive software development	1	10.0%
Kanban	3	30.0%
For Contracting development teams		
Extreme programming (XP)	3	25.0%
Scrum	5	41.7%
Adaptive software development	2	16.7%
Kanban	2	16.7%

5.8 Hybrid Agile Methods

Table 23 shows 48.3% of the respondents said that they had adopted a hybrid (software development practices adopted from more than a single agile method) agile method. The SCRUM (82.1%) method and XP (64%) methods are the two most popular ones for practices to create a hybrid method. The SCRUM method provides effective planning practices whereas the XP method practices mostly focus on implementation tasks. Hence, these two methods provide practices for an effective hybrid method. Interesting to see User-Centered Design (UCD) (38%) as part of a hybrid approach as, knowing the various user groups of the product, more reliable and achievable product backlogs can be created through the design phase of the project.

Table 23. Hybrid Agile Methods

Hybrid Agile Methods	Responses	Percentage
Extreme programming (XP)	18	64.3%
Scrum	23	82.1%
Adaptive software development	8	28.6%
Dynamic systems development method	4	14.3%

Feature-driven development	1	3.6%
Rapid application development	5	17.9%
Lean software development	6	21.4%
Kanban	12	42.9%
Agile modelling	2	7.1%
User-Centered Design (UCD)	8	28.6%
Total Responses	**28**	**48.3%**

Table 24. Hybrid Agile Methods Break-down

Hybrid Agile Methods	**Responses**	**Percentage**
For In-house development teams		
Extreme programming (XP)	6	54.5%
Scrum	8	72.7%
Adaptive software development	5	45.5%
Dynamic systems development method	2	18.2%
Feature-Driven Development (FDD)	1	9.1%
Rapid application development	3	27.3%
Lean software development	2	18.2%
Kanban	4	36.4%
User-Centered Design (UCD)	3	27.3%
Total Responses	**11**	**39.3%**
For Software vendors		
Extreme programming (XP)	12	80.0%
Scrum	13	86.7%
Adaptive software development	5	33.3%
Dynamic systems development method	1	6.7%
Lean software development	4	26.7%
Kanban	8	53.3%
Agile modelling	2	13.3%
User-Centered Design (UCD)	4	26.7%
Total Responses	**15**	**53.6%**
For Contracting development teams		
Scrum	2	100.0%
Adaptive software development	2	100.0%
Dynamic systems development method	1	50.0%
Rapid application development	2	100.0%
Agile modelling	1	50.0%

Total Responses	**2**	**7.1%**

5.9 Role to Compile Vision or Roadmap Plans

Table 25 highlights some of the new and emerging positions that are playing prominent roles in agile software development. The product manager role has now become crucial to provide direction with software product development – 69% of respondents indicated that a product manager is part of their team that compiles vision or roadmap plans. The product manager has substantial product domain knowledge and client business knowledge, including having foresight for future client needs, hence, they are in the best position to provide product vision and understanding at software engineering level.

Almost 66% of respondents have indicated that project managers help to compile vision/roadmap plans. This shows that there is a shift in the traditional roles and responsibilities of the project manager role. Project managers now are increasingly having to be multi-skilled (generalised skilled) to be part of the team to help compile vision/roadmap plans.

Table 25 also shows that almost 33% of respondents indicated that their agile development environment has a product development manager as part of the team to compile vision/roadmap plans. This is a new role that replaces the project manager role and is a proxy product manager role which is co-located with the development team managing and directing implementation in agile development teams (Lal, 2011).

In addition, Table 25 also shows that 29.30% of respondents have indicated that software engineering managers play a part in helping to compile vision/roadmap plans. Hence, the software engineering manager role too is a generalist type of role whereby engineering managers not only help develop reliable and achievable product vision/roadmap plans but also help to communicate it at a development team level so that they can swiftly develop effective and reliable product backlog/sprint plans.

Table 25. Role to Compile Vision or Roadmap Plans

Role to Compile Vision or Roadmap plans	Responses	Percentage
Product manager	40	69.0%
Project manager	38	65.5%
Business analyst	26	44.8%
Product development manager	19	32.8%
Software engineering manager	17	29.3%

Functional manager	10	17.2%
Marketing manager	5	8.6%

5.10 Measurement Model Validation

This section assesses the reliability and validity of the model. The measurement results enable the researcher to compare the theoretical measurement with the structural model and collected data. The reliability of the model can be assessed using an internal consistency reliability technique, while the validity of the model can be assessed using convergent validity and discriminant validity.

Internal Consistency Reliability

The first criteria to evaluate PLS is to test the Internal Composite Reliability (ICR). The result is obtained through the reliability of each latent variable. Reliability ensures that the portion of items used to measure a construct are related to be considered as a set of items. The reliability of a construct is calculated separately and assessed independently based on internal consistency (Netemeyer et al., 2003). The recommended value may vary between 0 and 1 with a minimum value of 0.7. Any higher value means that there is a higher reliability (Hair et al., 2011). Table 26 shows that all latent variables (agile organizations concepts) in the PLS model have internal consistencies greater than 0.7, indicating the reliability of all the constructs (agile organizations concepts) of this study.

Table 26. Composite Reliability Results

Construct	Composite Reliability
Competencies	0.842
Knowledge Management	0.851
Organizational Culture	0.815
Organizational Learning	0.836
Responsiveness	0.739
Speed	0.838
Team Effort	0.855
Workforce Agility	0.749

Convergent Validity

Convergent validity measures the positive correlation between each indicator with different indicators in the same construct. The result can be obtained through either

considering the outer loading of the indicators or the Average Variance Extracted (AVE). According to Hair et al. (2011), the AVE value should be greater than 0.50 in order to be accepted. Table 27 provides the AVE result, calculated using the statistical analysis tool SmartPLS, which suggests that the constructs used in this study all have convergent validity, confirming the relationship between the indicators of each construct.

Table 27. Convergent Reliability Results

Construct	AVE
Competencies	0.774
Knowledge Management	0.857
Organizational Culture	0.724
Organizational Learning	0.618
Responsiveness	0.760
Speed	0.768
Team Effort	0.797
Workforce Agility	0.605

Discriminant Validity

Discriminant validity determines how constructs are distinct from each other, compared by calculating the square root of the AVE values with the latent variable correlations. The recommended AVE value must be higher than the corresponding correlations among the latent variables (Hair et al., 2016). Table 28 shows the outer loadings and cross loadings of model constructs.

Table 28. Discriminant Validity Results

Construct	COM	KM	OC	OL	RES	SPD	TE	WA
Competencies (COM)	**0.807**							
Knowledge Management (KM)	0.529	**0.860**						
Organizational Culture (OC)	0.640	0.549	**0.774**					
Organizational Learning (OL)	0.670	0.661	0.629	**0.798**				
Responsiveness (RES)	0.479	0.648	0.672	0.658	**0.747**			
Speed (SPD)	0.656	0.549	0.542	0.712	0.475	**0.804**		
Team Effort (TE)	0.287	0.463	0.540	0.443	0.668	0.303	**0.823**	
Workforce Agility (WA)	0.493	0.611	0.501	0.616	0.572	0.501	0.419	**0.736**

5.11 Structural Model Validation

This section measures the effect based on sets of dependence relationships in the proposed model, providing an indication of the quality of the predictions. The outputs of PLS-SEM result are shown below in Figure 2

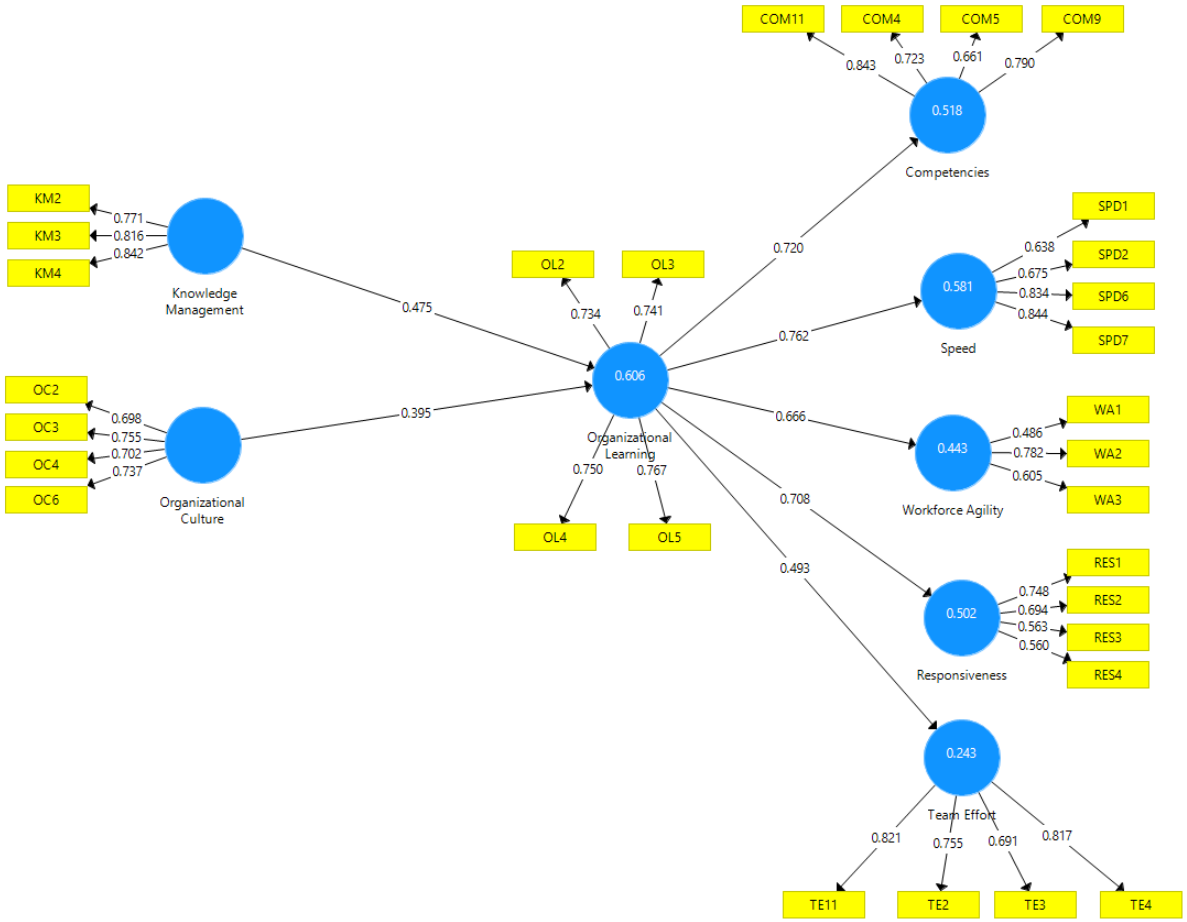


Figure 2. PLS-SEM proposed model results

The results of the validation test suggest that the proposed model passed all the criteria for assessment in relevance with the path coefficient and model loadings or weights on their measured latent variables. In addition, the formative measurement models are established properly with data from convergent validity (Table 27).

The PLS bootstrap procedure was recommended for this process. This process involves a large number of additional samples taken from the original sample at random and replaced with another when an observation is drawn. According to Hair et al. (2011), 5000 bootstrap samples are recommended for the PLS bootstrap procedure.

5.12 Coefficient of Determination

The most common method to evaluate the structural model is by using the coefficient of determination (R^2 value). The coefficient of determination measures the accuracy of the proposed model and allows comparison between the correlation of a specific construct's actual and predicted value (Hair et al., 2011). The value range for R^2 is between 0 and 1, where a value over 0.2 can be considered as high when measuring any specific thing. However, the R^2 must be at least 0.75 or higher when assessing customer satisfaction or loyalty (Hair et al., 2011). Table 29 provides the structural model's R^2 values.

Table 29. Coefficient of Determination (R^2 Values) Results

Construct	R^2 Value
Competencies	0.618
Organizational Learning	0.706
Responsiveness	0.602
Speed	0.681
Team Effort	0.643
Workforce Agility	0.537

Table 29 provides R^2 values showing the strongest relationship is provided by Organizational Learning (0.706), followed by Speed (0.681), then Team Effort (0.643), next is Competencies (0.618), then Responsiveness (0.602), and finally Workforce Agility which has the lowest value. The R^2 values provide a measurement of how the outcomes are replicated in the model which means organizational learning is the most critical organizational concept (Hair et al., 2016).

Table 30. Bootstrap Path Coefficient and T-Values in the Structural Model

Hypothesis	Path	Original Sample (O)	Sample Mean (M)	Standard Deviation (STDEV)	T-Values	P-Values
H1	**KM→ OL**	0.475	0.448	0.142	3.348	0.001
H2	**OC → OL**	0.395	0.420	0.103	3.816	0.000
H3	**OL → COM**	0.720	0.724	0.070	10.242	0.000
H4	**OL → RES**	0.708	0.727	0.080	8.819	0.000
H5	**OL → SPD**	0.762	0.763	0.070	10.908	0.000
H6	**OL → TE**	0.493	0.503	0.170	2.893	0.004
H7	**OL → WA**	0.666	0.661	0.113	5.904	0.000

The column, Original Sample (O), in Table 30 is used to determine how strong the relationship is between the hypothesized agile organization concepts in the structural model. According to Hair et al. (2011), the path coefficient values must range between 0 and 1 and only then can it be determined if the relationship is very weak (0 – 0.2), weak (0.2 – 0.4), moderate (0.4 – 0.6), strong (0.6 – 0.8) or very strong (0.8 – 1.0).

- **Hypothesis 1: Knowledge Management and Organizational Learning**

Table 30 shows a moderate relationship between knowledge management and organizational learning (Path = 0.475, t = 3.348, p<0.001). These results show that H1 is supported.

Hence, based on the agile concepts of Knowledge Management and Organizational Learning, Table 31 and Table 32 show the critical agile practices (from the survey results) that must be part of the agile software development environment regardless of agile values, principles and agile method practices for gaining agility.

Table 31. Key Agile Software Development Practices for Knowledge Management

Knowledge Management as part of Agile practices	Percentage
Collective (team) effort on all the tasks relating to product and project planning, implementation (development and testing).	94.7%
Task sharing at development level (including implementation).	91.4%
Work (method) practices at development level must help to build knowledge for future projects.	86.2%

Table 32. Key Agile Software Development Practices for Organizational Learning

Organizational Learning as part of Agile practices	Percentage
The (your) software engineering level has well thought-out practices and structures (functional unit and roles) based on continuous reviews and reflections.	91.4%
The learning and understanding on the organizational core competency has enabled the (your) development team to identify and learn firm-specific development competencies.	89.7%
The team-effort, ownership and cross-functional cooperation in the (your) development team are enforced through adoption of practices mutually accepted by all the stakeholders.	93.1%
The (your) development team is also driven by the mind set for continuous learning in projects.	91.4%

- **Hypothesis 2: Organizational Culture and Organizational Learning**

Table 30 shows a weak relationship between organizational culture and organizational learning (Path = 0.395, t = 3.816, p<0.000). These results indicate that H2 is also supported.

Hence, based on the agile concepts of Organizational Culture and Organizational Learning, Table 32 and Table 33 show the critical agile practices (from the survey results) that must be part of the agile software development environment regardless of agile values, principles and agile method practices for gaining agility.

Table 33. Key Agile Software Development Practices for Organizational Culture

Organizational Culture as part of Agile practices	Percentage
Face-to-face interaction and spontaneous collaboration are a critical culture in the (your) development team/organization.	89.7%
Individuals in the (your) team take part in a wide variety of work and decision-making.	89.6%
Collective decision-making is another critical element in the (your) development team/organization.	91.3%
Collective responsibility is a key part of the (your) development team's work practices.	89.7%

- **Hypothesis 3: Organizational Learning and Competencies**

Table 30 shows a strong relationship between organizational culture and organizational learning (Path = 0.720, t = 10.242, p<0.000). These results indicate that H3 is also supported.

Hence, based on the agile concepts of Organizational Learning and Competencies, Table 32 and Table 34 show the critical agile practices (from the survey results) that must be part of the agile software development environment regardless of agile values, principles and agile method practices for gaining agility.

Table 34. Key Agile Software Development Practices for Competencies

Competencies as part of Agile practices	Percentage
Continuous integration allows the (your) development team to fix most of the bugs before release.	87.9%

The (your) development team consists of individuals able to perform formal and informal (i.e. multiple) roles required in the project.	93.1%
The (your) development team has a fully integrated test-driven development environment.	84.5%
Test-driven development has enabled the (your) development team to deliver working software on a regular basis.	82.8%

- **Hypothesis 4: Organizational Learning and Responsiveness**

Table 30 shows a strong relationship between organizational culture and organizational learning (Path = 0.708, t = 8.819, p<0.000). These results indicate that H4 is also supported.

Hence, based on the agile concepts of Organizational Learning and Responsiveness Table 32 and Table 35 show the critical agile practices (from the survey results) that must be part of the agile software development environment regardless of agile values, principles and agile method practices for gaining agility.

Table 35. Key Agile Software Development Practices for Responsiveness

Responsiveness with Agile practices	Percentage
The empowerment of the (your) development team is a critical factor for the ability to learn and quickly adapt.	94.9%
The (your) development team has the ability to accept change in requirements during projects.	89.7%
The (your) development team has on the fly adopted or has the ability to adopt new development practices and skills.	94.8%
The (your) development team does beta releases of features to get useful insights and feedback to deliver useful features for the marketplace.	82.7%

- **Hypothesis 5: Organizational Learning and Speed**

Table 30 shows a strong relationship between organizational culture and organizational learning (Path = 0.762, t = 10.908, p<0.000). These results indicate that H5 is also supported.

Hence, based on the agile concepts of Organizational Learning and Speed Table 32 and Table 36 show the critical agile practices (from the survey results) that must be part of the agile software development environment regardless of agile values, principles and agile method practices for gaining agility.

Table 36. Key Agile Software Development Practices for Speed

Speed with Agile practices	Percentage
The (your) development team only works the normal hours per day or week on a consistent basis to deliver projects.	91.3%
Development infrastructure of the (your) development team does not place limitations to the team's expected work pace (speed).	82.8%
Avoiding technical debt is a critical mind set of (your) development team even if it affects the delivery speed of your team.	89.7%
TDD (Test-Driven Development) and refactoring are essential work practices to delivery software in short development cycles even though it affects the delivery speed of the (your) development team.	86.2%

- **Hypothesis 6: Organizational Learning and Team Effort**

Table 30 shows a moderate relationship between organizational culture and organizational learning (Path = 0.493, t = 2.893, p<0.004). These results indicate that H6 is also supported.

Hence, based on the agile concept of Organizational Learning and Team Effort Table 32 and Table 37 show the critical agile practices (from the survey results) that must be part of the agile software development environment regardless of agile values, principles and agile method practices for gaining agility.

Table 37. Key Agile Software Development Practices for Team Effort

Team Effort with Agile practices	Percentage
Product manager's collaboration with the (your) development team for input and feedback is critical to make effective vision/roadmap decisions.	96.6%
In the (your) development team, a cross-functional effort is required to create a reliable product backlog.	93.1%
Reviewing the product backlog more than once during the project timeline with the key stakeholders enables the (your) development team to deliver strategic benefits.	91.4%
The (your) whole team participates to achieve reliable sprint plans.	94.8%

- **Hypothesis 7: Organizational Learning and Workforce Agility**

Table 30 shows a strong relationship between organizational culture and organizational learning (Path = 0.666, t = 5.904, p<0.000). These results indicate that H7 is also supported.

Hence, based on the agile concept of Organizational Learning and Workforce Agility Table 32 and Table 38 show the critical agile practices (from the survey results) that must be part of the agile software development environment regardless of agile values, principles and agile method practices for gaining agility.

Table 38. Key Agile Software Development Practices for Workforce Agility

Workforce Agility with Agile practices	Percentage
The (your) development team consists of individuals with appropriate skills and knowledge to carry out multiple tasks.	98.3%
The (your) development team consists of highly skilled and competent individuals.	96.6%
The (your) development team has negotiation capabilities and consensus behaviour ability to accept change, generate new ideas and accept new responsibilities.	91.4%

Chapter Six: Discussion

6.0 Chapter Overview

This section provides discussion on the hypotheses that were tested through the data collected from the survey method, including discussion on agile practices that ought to be part of agile software development for gaining agility.

Table 39 shows the results of the seven hypotheses that were tested and Table 40 presents a list of agility practices based on agile organization concepts that ought to be part of agile software development based on the agile organization concepts. The research question for this study is

"What are the software development agility practices based on agile organization concepts?"

Table 39. Results of Hypothesis Tests

	Hypothesis	Result
H1	**Knowledge management will have a positive influence on organizational learning**	**Supported**
H2	**Organizational culture will have a positive influence on organizational learning**	**Supported**
H3	**Organizational learning will have a positive influence on competencies**	**Supported**
H4	**Organizational learning will have a positive influence on responsiveness**	**Supported**
H5	**Organizational learning will have a positive influence on speed**	**Supported**
H6	**Organizational learning will have a positive influence on team effort**	**Supported**
H7	**Organizational learning will have a positive influence on workforce agility**	**Supported**

Table 40. Identified Agile Practices Based on Agile Organization Concepts

Agile Organization Concepts	Agile Practices
Knowledge Management	• Collective effort provides solid critical thinking and problem-solving ability in the (your) development team. • In the (your) development team, task sharing in projects is vital to build knowledge and experience. • In the (your) development team, work (method) practices also build experience and knowledge for undertaking future projects.
Organizational Culture	• Face-to-face interaction and spontaneous collaboration are a critical culture in the (your) development team/organization.

	• Individuals in the (your) team take part in a wide variety of work and decision-making. • Collective decision-making is another critical element in the (your) development team/organization. • Collective responsibility is a key part of the (your) development team's work practices.
Organizational Learning	• The (your) software engineering level has well thought-out practices and structures (functional unit and roles) based on continuous reviews and reflections. • The learning and understanding on the organizational core competency has enabled the (your) development team to identify and learn firm-specific development competencies. • The team-effort, ownership and cross-functional cooperation in the (your) development team are enforced through adoption of practices mutually accepted by all the stakeholders. • The (your) development team is also driven by the mind-set for continuous learning in projects.
Competencies	• Continuous integration allows the (your) development team to fix most of the bugs before release. • The (your) development team consists of individuals able to perform formal and informal (i.e. multiple) roles required in the project. • The (your) development team has a fully integrated test-driven development environment. • Test-driven development has enabled the (your) development team to deliver working software on a regular basis.
Responsiveness	• The empowerment of the (your) development team is a critical factor for the ability to learn and quickly adapt. • The (your) development team has the ability to accept change in requirements during projects. • The (your) development team has on the fly adopted or has the ability to adopt new development practices and skills. • The (your) development team does beta releases of features to get useful insights and feedback to deliver useful features for the marketplace.
Speed	• The (your) development team only works the normal hours per day or week on a consistent basis to deliver projects. • Development infrastructure of the (your) development team does not place limitations to team's expected work pace (speed). • Avoiding technical debt is a critical mind set of (your) development team even if it affects the delivery speed of your team. • TDD (Test-Driven Development) and refactoring are essential work practices to delivery software in short development cycles even though it affects the delivery speed of the (your) development team.
Team Effort	• Product manager's collaboration with the (your) development team for input and feedback is critical to make effective vision/roadmap decisions. • In the (your) development team, a cross-functional effort is required to create a reliable product backlog. • Reviewing the product backlog more than once during the project timeline with the key stakeholders enables the (your) development team to deliver strategic benefits. • The (your) whole team participates to achieve reliable sprint plans
Workforce Agility	• The (your) development team consists of individuals with appropriate skills and knowledge to carry out multiple tasks. • The (your) development team consists of highly skilled and competent individuals.

	• The (your) development team has negotiation capabilities and consensus behaviour ability to accept change, generate new ideas and accept new responsibilities.

6.1 Discussion

H1 - Knowledge management will have a positive influence on organizational learning.

This hypothesis (H1) is supported for gaining agility in software development.

The investigation validates that this hypothesis (H1) is true (supported) for agile software development. To gain agility in software development, **knowledge management** is critical. This simply means that agile software development must have the mind set and be driven by knowledge management just like the agile software mind set for team effort, self-organising teams and cross-functional support (Chandani et al., 2007; Dove, 1999; Gupta et al., 2000; Pérez-Bustamante, 1999; Walczak, 2005). Knowledge (information, experience, understanding and skills) must be acquired through and on each and every method fragment (roles, practices and techniques) (Dove, 1999) when applied to carry out the work in agile software development projects.

A collection of method fragments make up a software development method or process (Chandani et al., 2007). In agile software development little knowledge is captured through documentation but most is tacit knowledge (Lei, Hitt, & Bettis, 1996) which the teams must build and enhance continuously. Tacit knowledge builds the transactive memory – a team memory system for collection and retrieval of development and business domain knowledge that helps to swiftly carry out work and effectively solve problems (Wegner, 1987). The knowledge management in agile software development is on the collective effort and shared tasks, these team-based practices not only allowing to undertake the project work to deliver the expected output swiftly but also ensuring the entire team becomes better (enhance) with skills and knowledge development (Table 40). Hence, agile software development projects ought to be driven by these practices to so that teams within have the knowledge and skills and are able to draw upon past experiences to carry out tasks on the next project successfully and achieve agility with software development.

Hence, to have the mind set for knowledge management drives the conviction for continuously organizational learning in the agile software development environment when carrying out the project work. Through continuous organizational learning well

thought-out practices and structures (functional unit and roles), i.e. method fragments, are achieved (hence reviews and reflections become part and parcel of development cycles in projects to become better for next cycle). It helps to understand organizational core competency to identify and learn firm-specific development competencies and has practices that are mutually accepted by all the stakeholders (Table 40).

These are all software development necessities in a market-driven environment to have software development agility. Well thought-out practices mean no bottlenecks and having nimble work practices enables swift ability to carry out work as expected. Focusing on the firm's core competency means the organization continuously becomes better at identifying and delivering new software features and products, which are highly innovative and bring high value automation for clients and customers. Continuously learning each other's (all stakeholders impacted by the project and product) needs and requirements helps to adopt and adapt mutually acceptable processes and practices that enable an effective cross-functional effort; the resulting output benefits the entire organization, functional team roles and clients/customers.

H2 - Organizational culture will have a positive influence on organizational learning

This hypothesis (H2) is supported for gaining agility in software development.

Organizational culture enables mutual benefits that provide competitive advantage at the marketplace (Janićijević, 2013). The culture in an agile environment requires team members to proactively to take rewarding action (Wendorff, 2002). The organizational culture increases the motivation for learning to enable flexibility with roles, functional units, processes and practices (Van Veelen et al., 2006). An agile culture enables quick responses for developing high value products through having dynamic functional units, roles, tasks, responsibilities and management (Christopher & Towill, 2000).

The investigation reveals that software development requires an organizational culture for face-to-face interaction and spontaneous collaboration, required to take part in a wide variety of work and decision-making, collective decision-making and collective responsibility. Hence the agile culture means enormous opportunity to learn and get better with software development for the benefit of the entire organization and clients/customers. Face-to-face interaction and spontaneous collaboration in a development environment allows individuals to probe in-depth to understand the

requirements, codes or any other technical knowledge from people who have that knowledge and understanding. Taking part in a wide variety of work in the software development environment allows continuous learning and understanding including getting knowledge and skills of other tasks, practices, processes and roles. Collective decision-making and responsibility means having the ability to make first time right decisions including ensuring a collective effort to implement the decision that was made in the software development environment. It is crucial to build an agile culture where team members solve problems better (Tolfo, Wazlawick, Ferreira, & Forcellini, 2011).

Hence, an agile culture is critical for organizational learning for well thought-out practices and structures (functional units and roles) (i.e. method fragments); understanding on the organizational core competency and for mutually accepted practices by all the stakeholders. Without these organizational learnings becoming better, agility will not be there to identify and deliver market-driven software features and products ahead of the competition. Hence, an agile organizational culture and learning must drive an agile software development environment.

H3 - Organizational learning will have a positive influence on competencies.

This hypothesis (H3) is supported for gaining agility in software development.

Test-driven development, continuous integration and individuals to perform in formal and informal (i.e. multiple) roles in agile software development teams, are critical and ought to be part of core competencies for agility in the product development environment. According to Plonka (1997), agility requires individuals to take up additional responsibilities. Hence, with software development it is through informal roles which individuals must perform in while having a formal role. Informal roles are temporary and can be for a sprint or for a project only. These agile competencies require continuous reflection for learning for improvement from one short development cycle to another throughout the project so that development teams have relevant and effective practices to meet current goals and objectives for the business organization.

Without the competency for test-driven development and continuous integration there is no possibility to deliver working software in short development cycles. Having individual competency in software development teams to step up into formal and informal software engineering roles (be a software engineer but at the same time perhaps setup as team leader or technical leader for a sub-team) is important in an agile setup to have

sufficient development capacity to be able to deliver software in short development cycles. For individuals to work in formal and informal roles requires continuous learning on the fly for the required skills through task sharing with another individual or with the entire team. Hence, on the fly learning must drive all agile software development practices and roles for any software development to be able to gain agility.

Core competencies deal with the ability of an organization to develop unique business practices and the ability to respond quickly to market opportunities and competitive threats (Assen, 2000). Employees must have the speed to develop new skills, obtain necessary skills to perform changes, innovative management skills and obtain IT skills (software and hardware) in order to become more competent (Breu et al., 2002). Software development agility depends upon competent development teams (Dybå & Dingsøyr, 2008; Hedberg, 2015).

H4 - Organizational learning will have a positive influence on responsiveness

This hypothesis (H4) is supported for gaining agility in software development.

The following practices define the responsive abilities in agile software development: empowerment to learn and quickly adapt, ability to accept changes in requirements, ability on the fly to adopt new development practices and skills and ability for beta releases to get useful insights and feedback (learn) to deliver useful features for the marketplace. These practices are critical and must drive the agile software development environment since responsive ability defines agility capability.

According to (Sharifi & Zhang, 1999), this ability in agile organizations enables them to continuously anticipate, detect and deal with changes, including being able to swiftly plan for an immediate response. In addition, organizational learning ability is critical to enable responsive ability (Cohen & Levinthal, 1990; Schulz, 2001). Hence, agility with software development requires responsive teams and developers with high skills in order to create change and respond rapidly to change in an uncertain environment (Cockburn & Highsmith, 2001).

For responsive ability in agile software development, all the work practices relating to creating and managing product backlogs including the short development cycles (others as well) requires continuous learning to adapt and build skills and understanding to deliver projects incorporating the changing needs of all critical stakeholders. In addition, successful responsive ability is also defined by ability to learn

and improve through feedback on implemented features from potential users. Making sure the practices remain reliable from one project to another is critical for development agility to able to meet successful market release. Hence, the mind set for organizational learning is vital for having responsive ability with agile software development.

H5 - Organizational learning will have a positive influence on speed

This hypothesis (H5) is supported for gaining agility in software development.

Agility is also dependent upon speed, which relates to in-house capability to get work done on time, enabling competitive advantage at the marketplace (Gandossy, 2003). It is no different for agile software development where speed is critical to be on time at the marketplace with outstanding software. Speed enabling agility with software development means the in-house capability for making software features and products available rapidly, faster than competitors (Duguay, Landry, & Pasin, 1997). Hence, practices that ought to be part of agile software development for agility gains are as follows: (1) working normal hours per day or week on a consistent basis (avoid burnout so that team's performance is not affected in projects); (2) no limitations to a team's expected work pace (speed); (3) The pace of work must be based on practices that avoid technical debt, and TDD (Test-Driven Development) and refactoring are essential work practices and must help to determine the work pace.

However, speed is dependent upon well-thought practices (no bottle necks), solid technical understanding and skills (core competency), a team effort based on cross-functional support and collaboration enabling swift designing making including on-going reflection for learning throughout the project. Hence, on-going organizational learning on these practices is vital as it impacts speed or pace of work.

Speed enables agile organizations to respond to unexpected changes, deliver products and services rapidly to the customers and provide a competitive advantage for the agile organization in time-based competition (Sharifi & Zhang, 1999).

H6 - Organizational learning will have a positive influence on team effort

This hypothesis (H6) is supported for gaining agility in software development.

Agility in software development is achieved through collective cross-functional team effort and ownership which involves business function at the development level.

The team approach enables delivering quality products in the shortest time (Edmondson & Nembhard, 2009). This enables agility capability for their market-driven product development (Sherehiy et al., 2007). In agile software development, team effort enables coordination of the domain and technical knowledge for swift development and availability of new features (Faraj & Sproull, 2000).

The practices that ought to be part of agile software development for agility are as follows (Table 40): for product planning driven by business (product managers) collaboration with the development team for input and feedback, project planning done through a cross-functional effort to create a reliable product backlog, product backlog reviews involving key stakeholders and a collective team effort to achieve reliable sprint plans.

However, the team effort is also dependent upon organization learning. It requires continuous learning on team effort to have well-thought practices that enables effective team effort, solid technical understanding and skills (core competency) for high team performance, and practices that are based on cross-functional support and collaboration.

Collective effort from development teams is used to make decisions on various tasks (task sharing) rather than based on role or individual effort (Yauch, 2007). Organizational learning allows employees to obtain necessary knowledge when carrying out tasks. The tacit knowledge is often generated and exchanged during collaboration for problem solving (Lam, 2000). In a collaborative environment, employees maintain mutually shared cognition which leads to an increase in team performance

H7 - Organizational learning will have a positive influence on workforce agility

This hypothesis (H7) is supported for gaining agility in software development.

Workforce agility capability enables product management individuals to swiftly adapt to changes through acquiring new technical skills, new business knowledge and understanding, and modern process and practices to deal with unpredictable changes and competition in the market (Qin & Nembhard, 2015; Sohrabi et al., 2014). Therefore, agile organizations ought to have a highly skilled and adaptable workforce in order to have the flexibility capability and to have ability to successfully deal with unstructured tasks as a result of competition (Youndt et al., 1996). According to Santos-Vijande, López-Sánchez, and Trespalacios (2012), organizational learning gives organizations the ability to adapt and develop successful products in evolving market conditions. Workforce agility means

an agile software development environment has competent engineers with responsive behaviour capable of swift problem-solving related to development and having decision-making abilities (Plonka, 1997). Engineers also adapt with responsive behaviour to deliver features according to new priorities (market or business objective changes).

Without workforce agility, software development agility will not be achieved. Hence for software development agility, agile practices must drive the product development environment. Development teams must consist of individuals with appropriate skills and knowledge to carry out multiple tasks including having highly skilled and competent individuals. All software development practices must be based around the ability to negotiate, accept changes, generate new ideas and accept new responsibilities. All these are achieved through the mind set for organizational learning whereby the development teams identify and learn firm-specific development competencies.

Chapter Seven: Conclusion

This study investigated agile software development practices based on agile organisation concepts to provide an understanding on agility with software development. Agile organisation concepts are critical factors leading to agility regardless of the type of business. However, it is not known if agile organisation concepts are also the basis for adopting agile methods and their practices for software development. With agile software development, agile values and principles (agile manifesto) ought to drive agile adoption, which may lead to agility. Hence, this investigation's findings clearly identify agile practices which will aid and support agile values and principles for successful agile method adoption for achieving agility with software development.

The aim of this study was to learn directly from the software development teams to provide a list of agile method practices together with agile values and principles that drive agile method adoption that lead to agility in software development. Agility in software development is crucial due to the dynamic nature of business and its requirements, impacted by the current business environment, which is market-driven with customer and competitive challenges, and emerging technologies. Agility with software development enables appropriate software development method practices able to deliver software that matches businesses and their end-users' needs. Hence, this investigation with its findings, i.e. agile software development practices based on agile organisation concepts, will aid and support agile values and principles for successful agile adoption for achieving agility with software development.

This study involved a quantitative approach using a web-based survey to collect data. The web-based survey allowed targeting of a wide-range of participants from the software development community. An exploratory study was employed to investigate and identify the agile software development practices that lead to agility based on the agile organization concepts. The agile organization concepts and agile software development practices were identified through the literature review.

The research question was *"What are the software development agility practices based on agile organization concepts?"*

To answer the research question, first, a literature review was undertaken to identify agile organization concepts. Eight agile organisation concepts were identified: knowledge management, organizational culture, organizational learning, competencies, responsiveness, speed, team effort and workforce agility. Second, a literature review on

agile software development practices was done to identify key agile method practices. Third, the agile method practices were matched with specific agile organization concepts and used to construct the survey questionnaires. Finally, based on agile organisation concepts, seven hypotheses were identified to be tested to identify the agile software development practices for agility based on the survey that was undertaken.

Initially when the survey questions were designed, there were 120 questions but after feedback and improvement from supervisor, 67 questions were selected reflecting the eight agile organization concepts. The researcher pre-tested the survey questions by involving a master's student also researching agile software development and two other Auckland University of Technology lecturers who are experts in agile software development.

The potential participants were identified through agile community groups and various agile software development conferences held in New Zealand and Australia. Before conducting the web-based survey, the researcher was given ethical approval by the ethics committee to conduct the research.

The survey invitations were sent to software development organizations or teams in New Zealand and Australia. 181 invitations were emailed to potential participants (in-house agile software development teams, agile software vendors and agile software development contracting companies). 63 participants responded to the online-survey. From 63 responses, 58 met the criteria for PLS data analysis. From the results section, there were 19 responses from in-house development teams, 25 responses from software vendors, and 14 responses from contracting development teams.

For statistical analysis this study used the partial least squared (PLS) over covariance-based SEM (CB-SEM). The required data for CB-SEM is at least two hundred responses and is mostly used for theory testing or theory confirmation. The goal of this study was to identify key constructs which were suitable with exploratory research and the researcher used SmartPLS as the analysis tool.

The Table 30 provides the information that seven hypotheses values have been met based on agile organization concepts.

Table 30. Bootstrap Path Coefficient and T-Values in the Structural Model (repeated table)

Hypothesis	Path	Original Sample (O)	Sample Mean (M)	Standard Deviation (STDEV)	T-Values	P-Values
H1	**KM→ OL**	0.475	0.448	0.142	3.348	0.001
H2	**OC → OL**	0.395	0.420	0.103	3.816	0.000
H3	**OL → COM**	0.720	0.724	0.070	10.242	0.000
H4	**OL → RES**	0.708	0.727	0.080	8.819	0.000
H5	**OL → SPD**	0.762	0.763	0.070	10.908	0.000
H6	**OL → TE**	0.493	0.503	0.170	2.893	0.004
H7	**OL → WA**	0.666	0.661	0.113	5.904	0.000

T-value evaluates the difference relative to sample data from the standard error during hypothesis testing. When the t-value equals to 0, it means the sample results are exactly equal to the null hypothesis.

P-value measures the probability of the null hypothesis (opposite direction of the observed hypothesis) being true compared with confidence intervals. If the p-value is smaller than the confidence intervals, it means the null hypothesis is rejected and if the p-value is near the confidence interval, it indicates the null hypothesis is accepted and the current hypothesis is rejected. The confidence interval ranged from 0.01, 0.05 and 0.1. This study chose 0.05 because it was the most common confidence interval used in the quantitative study.

Table 29. R^2 Value Results (repeated table)

Construct	R^2 Value
Competencies	0.618
Organizational Learning	0.706
Responsiveness	0.602
Speed	0.681
Team Effort	0.643
Workforce Agility	0.537

The R^2 values provide measurement of how the outcomes are replicated in the model which means organizational learning is the most critical organizational concept (Hair et al., 2016). In addition, other agile organization concepts were met.

Table 41. Hypothesis Results

Hypothesis	Path	Conditions
H1	Knowledge Management → Organizational Learning	Supported
H2	Organizational Culture → Organizational Learning	Supported
H3	Organizational Learning → Competencies	Supported
H4	Organizational Learning → Responsiveness	Supported
H5	Organizational Learning → Speed	Supported
H6	Organizational Learning → Team Effort	Supported
H7	Organizational Learning → Workforce Agility	Supported

Table 40 explains the agile organization concepts and related agile practices that were identified from respondents with SmartPLS.

Table 40. Identified Agile Practices Based on Agile Organization Concepts (repeated table)

Agile Organization Concepts	Agile Practices
Knowledge Management	• Collective effort provides solid critical thinking and problem-solving ability in the (your) development team. • In the (your) development team, task sharing in projects is vital to build knowledge and experience. • In the (your) development team, work (method) practices also build experience and knowledge for undertaking future projects.
Organizational Culture	• Face-to-face interaction and spontaneous collaboration are a critical culture in the (your) development team/organization. • Individuals in the (your) team take part in a wide variety of work and decision-making. • Collective decision-making is another critical element in the (your) development team/organization. • Collective responsibility is a key part of the (your) development team's work practices.
Organizational Learning	• The (your) software engineering level has well thought-out practices and structures (functional unit and roles) based on continuous reviews and reflections. • The learning and understanding on the organizational core competency has enabled the (your) development team to identify and learn firm-specific development competencies. • The team-effort, ownership and cross-functional cooperation in the (your) development team are enforced through adoption of practices mutually accepted by all the stakeholders. • The (your) development team is also driven by the mind-set for continuous learning in projects.
Competencies	• Continuous integration allows the (your) development team to fix most of the bugs before release. • The (your) development team consists of individuals able to perform formal and informal (i.e. multiple) roles required in the project. • The (your) development team has a fully integrated test-driven development environment. • Test-driven development has enabled the (your) development team to deliver working software on a regular basis.
Responsiveness	• The empowerment of the (your) development team is a critical factor for the ability to learn and quickly adapt. • The (your) development team has the ability to accept change in requirements during projects. • The (your) development team has on the fly adopted or has the ability to adopt new development practices and skills. • The (your) development team does beta releases of features to get useful insights and feedback to deliver useful features for the marketplace.
Speed	• The (your) development team only works the normal hours per day or week on a consistent basis to deliver projects. • Development infrastructure of the (your) development team does not place limitations to team's expected work pace (speed).

	• Avoiding technical debt is a critical mind set of (your) development team even if it affects the delivery speed of your team. • TDD (Test-Driven Development) and refactoring are essential work practices to delivery software in short development cycles even though it affects the delivery speed of the (your) development team.
Team Effort	• Product manager's collaboration with the (your) development team for input and feedback is critical to make effective vision/roadmap decisions. • In the (your) development team, a cross-functional effort is required to create a reliable product backlog. • Reviewing the product backlog more than once during the project timeline with the key stakeholders enables the (your) development team to deliver strategic benefits. • The (your) whole team participates to achieve reliable sprint plans
Workforce Agility	• The (your) development team consists of individuals with appropriate skills and knowledge to carry out multiple tasks. • The (your) development team consists of highly skilled and competent individuals. • The (your) development team has negotiation capabilities and consensus behaviour ability to accept change, generate new ideas and accept new responsibilities.

The agile organization concepts and their related agile practices can be used for software development teams or organizations as a guide to consider and develop their agile approach. By doing that, software development teams can measure their own capability to deliver working software efficiently and become more responsive to changes.

7.0 Theoretical Contributions

This thesis highlights and promotes an understanding of the importance of agility in the software industry with agile organization concepts and agile software development practices. The four agile values and twelve principles are also important because software development teams tend to forget them; thus, they cannot achieve absolute agility in real practice. Software development teams can become agile if they follow more transparency, evaluation and adaptation towards the software project. From transparency, software development teams can track the visibility and delivery process to the customers without affecting the project results. With frequent evaluation, software development teams can identify unacceptable traits that may bring critical failure to the development process and the adaptation allows software development team to adjust the development process rapidly and efficiently.

7.1 Practical Contributions

The practical implications from this investigation to achieve software development agility are as follows:

1. For knowledge management, the agile method practices must help to create the work break-down structure for a collective effect (team effort on important tasks such as product backlog and sprint planning, retrospectives) and task sharing (two

or more individuals collectively working on the same planning and engineering tasks and for knowledge creation and retention).

2. All agile method practices defining the tasks must be carried out through the organisation culture for face-to-face communication, including delivering through the organisational culture for collective decision-making and responsibility for planning, design and implementation.
3. For organisational learning, the agile method practices must also enforce continuous learning to enhance organisational culture, teamwork, technical skills, roles, business domain knowledge and development infrastructure or tools, including the adopted agile process and method practices.
4. The team must adopt and continuously adapt the agile method practices that enable in-house software development competencies for the Test-Driven Development (TDD) practices, and continuous code integration, including regular bug free software releases through short development cycles.
5. For responsiveness ability, the development team must adopt a mind-set for change, driven and empowered for adoption and adaptation of agile method practices.
6. For speed, agile method practices must help to build a consistent work pace for the entire project without team burnouts, must enforce an acceptable level of productivity supported by high productivity tools/infrastructure, while incorporating all the required practices to deliver high quality software products or features.
7. All agile method practices are based upon collective team effort.
8. For work force agility, agile method practices must also help to develop highly skilled and highly capable individuals for continuous learning.

7.2 Limitations

The limitations of the study are as follows:

First, the respondents were mostly from New Zealand and a few from Australia. Thus the responses may vary when compared with software development organizations or teams in other countries.

Second, the survey had a response rate of 32% from the 181 invitations sent to software development organizations or teams. A better response rate would have provided a better understanding on agile organisation concepts and the related practices.

Third, there was no response related to the scaled agile method in the survey. The researcher hopes it can be a topic for future research.

7.3 Future Research

This study serves as foundation for future research which may investigate agile organisation concepts through qualitative study to do an in-depth investigation with a few selected case study organisations to identify and provide a list of agile software development practices for agility.

A future survey could be carried-out for world-wide participation through collaboration with researchers from the other countries which could help to validate the results of this study.

References

Abrahamsson, P., Conboy, K., & Wang, X. (2009). 'Lots done, more to do': the current state of agile systems development research: Springer.

Agarwal, N., Karimpour, R., & Ruhe, G. (2014). Theme-based product release planning: An analytical approach*IEEE.* Symposium conducted at the meeting of the System Sciences (HICSS), 2014 47th Hawaii International Conference on

Agile Manifesto. (2001). *Manifesto for agile software development*. Retrieved from http://www.agilemanifesto.org/

Allwood, J. M., Childs, T. H. C., Clare, A. T., De Silva, A. K. M., Dhokia, V., Hutchings, I. M., . . . Turner, S. (2016). Manufacturing at double the speed. *Journal of Materials Processing Technology, 229*, 729-757. doi:http://dx.doi.org/10.1016/j.jmatprotec.2015.10.028

Alqudah, M., & Razali, R. (2016). A Review of Scaling Agile Methods in Large Software Development.

Ambler, S. W., & Lines, M. (2012). *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*: Pearson Education. Retrieved from https://books.google.co.nz/books?id=CwvBEKsCY2gC

Anderson, J. C., & Gerbing, D. W. (1988). Structural equation modeling in practice: A review and recommended two-step approach. *Psychological bulletin, 103*(3), 411.

Anderson, L., Alleman, G. B., Beck, K., Blotner, J., Cunningham, W., Poppendieck, M., & Wirfs-Brock, R. (2003). Agile management-an oxymoron?: who needs managers anyway?*ACM.* Symposium conducted at the meeting of the Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications

Andrews, D., Nonnecke, B., & Preece, J. (2007). Conducting research on the internet:: Online survey design, development and implementation guidelines.

Antanovich, A., Sheyko, A., & Katumba, B. (2010). *Bottlenecks in the Development Life Cycle of a Feature-A Case Study Conducted at Ericsson AB*.

Antonacopoulou, E., & Chiva, R. (2007). The social complexity of organizational learning: the dynamics of learning and organizing. *Management Learning, 38*(3), 277-295.

Anwar, S., Motla, Y. H., Siddiq, Y., Asghar, S., Hassan, M. S., & Khan, Z. I. (2014, 8-10 Dec. 2014). User-centered design practices in scrum development process: A distinctive advantage? Symposium conducted at the meeting of the 17th IEEE International Multi Topic Conference 2014 doi:10.1109/INMIC.2014.7097330

Appelbaum, S. H., & Gallagher, J. (2000). The competitive advantage of organizational learning. *Journal of Workplace Learning, 12*(2), 40-56. doi:doi:10.1108/13665620010316000

Arechavala-Vargas, R., Diaz-Perez, C., Madrigal-Torres, B. E., & Ferrer-Ramirez, S. (2007, 5-9 Aug. 2007). Organizational Learning Strategies and Managerial Culture in Software Firm Networks in Mexico Symposium conducted at the meeting of the PICMET '07 - 2007 Portland International Conference on Management of Engineering & Technology doi:10.1109/PICMET.2007.4349419

Ashmore, S., & Runyan, K. (2014). *Introduction to Agile Methods*: Pearson Education. Retrieved from https://books.google.co.nz/books?id=hE7iAwAAQBAJ

Assen, M. F. v. (2000). Agile-based competence management: the relation between agile manufacturing and time-based competence management. *International Journal of Agile Management Systems, 2*(2), 142-155. doi:doi:10.1108/14654650010337168

Babar, M. A., Brown, A. W., & Mistrik, I. (2013). *Agile Software Architecture: Aligning Agile Processes and Software Architectures*: Elsevier Science. Retrieved from https://books.google.co.nz/books?id=wJb9pG1gcUsC

Bardis, P. D. (1979). Social Interaction and Social Processes. *Social Science, 54*(3), 147-167.

Barney, J. B. (1986). Organizational culture: can it be a source of sustained competitive advantage? *Academy of management review, 11*(3), 656-665.

Barton, B., & Brooks, B. (2012, 13-17 Aug. 2012). Agile's Role in Developing Robust Software Competency at Precor Symposium conducted at the meeting of the 2012 Agile Conference doi:10.1109/Agile.2012.29

Batra, D., Xia, W., VanderMeer, D., & Dutta, K. (2010). Balancing agile and structured development approaches to successfully manage large distributed software projects: A case study from the cruise line industry. *Communications of the Association for Information Systems, 27*(1), 21.

Beersma, B., Hollenbeck, J. R., Humphrey, S. E., Moon, H., Conlon, D. E., & Ilgen, D. R. (2003). Cooperation, competition, and team performance: Toward a contingency approach. *Academy of Management Journal, 46*(5), 572-590.

Bernardes, E. S., & Hanna, M. D. (2009). A theoretical review of flexibility, agility and responsiveness in the operations management literature: Toward a conceptual definition of customer responsiveness. *International Journal of Operations & Production Management, 29*(1), 30-53.

Billington, C., & Amaral, J. (1999). Investing in product design to maximize profitability through postponement. *Achieving Supply Chain Excellence Through Technology, San Francisco: Montgomery Research*.

Blackburn, J. D., Scudder, G. D., & Wassenhove, L. N. V. (1996). Improving speed and productivity of software development: a global survey of software developers. *IEEE Transactions on Software Engineering, 22*(12), 875-885. doi:10.1109/32.553636

Blaikie, N. (2009). *Designing Social Research*: Wiley. Retrieved from https://books.google.co.nz/books?id=lpCeLiDLZVYC

Boehm, B. (2002). Get ready for agile methods, with care. *Computer, 35*(1), 64-69.

Boehm, B. (2003). Value-based software engineering: reinventing. *ACM SIGSOFT Software Engineering Notes, 28*(2), 3.

Bollen, K., & Lennox, R. (1991). Conventional wisdom on measurement: A structural equation perspective. *Psychological bulletin, 110*(2), 305.

Bottani, E. (2009). On the assessment of enterprise agility: issues from two case studies. *International Journal of Logistics: Research and Applications, 12*(3), 213-230.

Bougroun, Z., Zeaaraoui, A., & Bouchentouf, T. (2014, 20-22 Oct. 2014). The projection of the specific practices of the third level of CMMI model in agile methods: Scrum, XP and Kanban Symposium conducted at the meeting of the 2014 Third IEEE International Colloquium in Information Science and Technology (CIST) doi:10.1109/CIST.2014.7016614

Braz, A., Rubira, C. M. F., & Vieira, M. (2015, 3-7 Aug. 2015). Development of Complex Software with Agile Method Symposium conducted at the meeting of the 2015 Agile Conference doi:10.1109/Agile.2015.18

Breu, K., Hemingway, C. J., Strathern, M., & Bridger, D. (2002). Workforce agility: the new employee strategy for the knowledge economy. *Journal of Information Technology, 17*(1), 21-31.

Brown, A. L., & Palincsar, A. S. (1989). Guided, cooperative learning and individual knowledge acquisition. *Knowing, learning, and instruction: Essays in honor of Robert Glaser*, 393-451.

Brown, A. W., Ambler, S., & Royce, W. (2013). *Agility at scale: economic governance, measured improvement, and disciplined delivery*. presented at the meeting of the Proceedings of the 2013 International Conference on Software Engineering, San Francisco, CA, USA.

Bryman, A. (2012). *Social Research Methods*: OUP Oxford. Retrieved from https://books.google.co.nz/books?id=vCq5m2hPkOMC

Burns, T., & Stalker, G. M. (1961). *The management of innovation*. [London: Tavistock Publications. Retrieved from /z-wcorg/ database.

Burns, T., & Stalker, G. M. (1994). *The Management of Innovation*: Oxford University Press. Retrieved from https://books.google.co.nz/books?id=SFev_VZDlIUC

Capretz, L. F. (2003). Personality types in software engineering. *International Journal of Human-Computer Studies, 58*(2), 207-214. doi:http://dx.doi.org/10.1016/S1071-5819(02)00137-4

Chandani, A., Neeraja, B., & Sreedevi. (2007, 20-22 Dec. 2007). Knowledge Management: An overview & its impact on software industry Symposium conducted at the meeting of the Information and Communication Technology in Electrical Sciences (ICTES 2007), 2007. ICTES. IET-UK International Conference on

Chen, G. (2005). Newcomer adaptation in teams: Multilevel antecedents and outcomes. *Academy of Management Journal, 48*(1), 101-116.

Chen, W., & Hirschheim, R. (2004). A paradigmatic and methodological examination of information systems research from 1991 to 2001. *Information systems journal, 14*(3), 197-235.

Chesebrough, P. H., & Davis, G. B. (1983). PLANNING A CAREER PATH IN INFORMATION-SYSTEMS. *Journal of Systems Management, 34*(1), 6-13.

Chin, W. (2000). Partial least squares for IS researchers: an overview and presentation of recent advances using the PLS approach Symposium conducted at the meeting of the ICIS

Chin, W. W., & Newsted, P. R. (1999). Structural equation modeling analysis with small samples using partial least squares. *Statistical strategies for small sample research, 2*, 307-342.

Chonko, L. B., & Jones, E. (2005). The Need for Speed: Agility Selling. *Journal of Personal Selling & Sales Management, 25*(4), 371-382. doi:10.1080/08853134.2005.10749071

Christopher, M. (2000). The Agile Supply Chain. Competing in Volatile Markets [Article]. *Industrial Marketing Management, 29*, 37-44. doi:10.1016/S0019-8501(99)00110-8

Christopher, M., & Towill, D. R. (2000). Supply chain migration from lean and functional to agile and customised. *Supply Chain Management: An International Journal, 5*(4), 206-213.

Chromatic. (2003). *Extreme Programming Pocket Guide*: O'Reilly Media. Retrieved from https://books.google.co.nz/books?id=Wt0FlVWrEXkC

Clancy, T. (2014). The Standish Group Report.

Cobb, C. G. (2015). *The Project Manager's Guide to Mastering Agile: Principles and Practices for an Adaptive Approach*: Wiley. Retrieved from https://books.google.co.nz/books?id=vHjTBQAAQBAJ

Cockburn, A. (2002). *Agile software development* (Vol. 177): Addison-Wesley Boston.

Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *34*(11), 131-133.

Codita, R. (2011). *Contingency Factors of Marketing-Mix Standardization: German Consumer Goods Companies in Central and Eastern Europe*: Gabler Verlag. Retrieved from https://books.google.co.nz/books?id=kewkSVcjp1sC

Cohen, L., Manion, L., & Morrison, K. (2013). *Research Methods in Education*: Taylor & Francis. Retrieved from https://books.google.co.nz/books?id=mLh0Oza3V1IC

Cohen, W. M., & Levinthal, D. A. (1990). Absorptive capacity: A new perspective on learning and innovation. *Administrative science quarterly*, 128-152.

Cohn, M. (2004). *User Stories Applied: For Agile Software Development (Adobe Reader)*: Pearson Education. Retrieved from https://books.google.co.nz/books?id=DHZP_YL3FxYC

Colomo-Palacios, R., Tovar-Caro, E., García-Crespo, Á., & Gómez-Berbís, J. M. (2012). Identifying technical competences of it professionals: The case of software engineers. *Professional Advancements and Management Trends in the IT Sector, 1*.

Coltman, T., Devinney, T. M., Midgley, D. F., & Venaik, S. (2008). Formative versus reflective measurement models: Two applications of formative measurement. *Journal of Business Research, 61*(12), 1250-1262.

Creswell, J. W. (2009). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*: SAGE Publications. Retrieved from https://books.google.co.nz/books?id=bttwENORfhgC
Creswell, J. W. (2013). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*: SAGE Publications. Retrieved from https://books.google.co.nz/books?id=EbogAQAAQBAJ
Crispin, L., & Gregory, J. (2008). *Agile Testing: A Practical Guide for Testers and Agile Teams*: Pearson Education. Retrieved from https://books.google.co.nz/books?id=68_IhPvoKS8C
Crotty, M. (1998). *The Foundations of Social Research: Meaning and Perspective in the Research Process*: SAGE Publications. Retrieved from https://books.google.co.nz/books?id=j4hXocGn1yIC
Danesh, A. S. (2011). A survey of release planning approaches in incremental software development. In *Computational Intelligence and Information Technology* (pp. 687-692): Springer.
Darroch, J. (2005). Knowledge management, innovation and firm performance. *Journal of Knowledge Management, 9*(3), 101-115. doi:doi:10.1108/13673270510602809
Dasgupta, M., & Gupta, R. (2009). Innovation in organizations: A review of the role of organizational learning and knowledge management. *Global Business Review, 10*(2), 203-224.
Dávideková, M., & Ml, M. G. (2016, 22-24 Aug. 2016). Software Application Logging: Aspects to Consider by Implementing Knowledge Management Symposium conducted at the meeting of the 2016 2nd International Conference on Open and Big Data (OBD) doi:10.1109/OBD.2016.22
DeClue, T. (2003). Pair programming and pair trading: effects on learning and motivation in a CS2 course. *The Journal of Computing Sciences in Colleges, 18*(5), 49-56.
Dodgson, M. (1993). Organizational Learning: A Review of Some Literatures. *Organization Studies, 14*(3), 375-394. doi:doi:10.1177/017084069301400303
Dömges, R., & Pohl, K. (1998). Adapting traceability environments to project-specific needs. *Communications of the ACM, 41*(12), 54-62.
Donaldson, L. (2001). *The Contingency Theory of Organizations*: SAGE Publications. Retrieved from https://books.google.co.nz/books?id=bbRhBAAAQBAJ
Donaldson, S. E., & Siegel, S. G. (2001). *Successful Software Development*: Prentice Hall PTR. Retrieved from https://books.google.co.nz/books?id=lrix5MNRiu4C
Dorairaj, S., Noble, J., & Malik, P. (2012, 13-17 Aug. 2012). Knowledge Management in Distributed Agile Software Development Symposium conducted at the meeting of the 2012 Agile Conference doi:10.1109/Agile.2012.17
Döscher, K. (2014). *Recovery Management in Business-to-Business Markets: Conceptual Dimensions, Relational Consequences and Financial Contributions*: Springer Fachmedien Wiesbaden. Retrieved from https://books.google.co.nz/books?id=7DtdAwAAQBAJ
Doshi, V. P., & Patil, V. (2016, 24-26 Feb. 2016). Competitor driven development: Hybrid of extreme programming and feature driven reuse development Symposium conducted at the meeting of the 2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS) doi:10.1109/ICETETS.2016.7602985
Dove, R. (1999). Knowledge management, response ability, and the agile enterprise. *Journal of knowledge management, 3*(1), 18-35.
Drury, M., Conboy, K., & Power, K. (2012). Obstacles to decision making in Agile software development teams. *Journal of Systems and Software, 85*(6), 1239-1254. doi:http://dx.doi.org/10.1016/j.jss.2012.01.058
Duffy, J. (2000). Knowledge management: What every information professional should know. *Information Management, 34*(3), 10.

Duguay, C. R., Landry, S., & Pasin, F. (1997). From mass production to flexible/agile production. *International Journal of Operations & Production Management, 17*(12), 1183-1195.
Duvall, P. M., Matyas, S., & Glover, A. (2007). *Continuous Integration: Improving Software Quality and Reducing Risk*: Pearson Education. Retrieved from https://books.google.co.nz/books?id=PV9qfEdv9L0C
Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology, 50*(9–10), 833-859. doi:http://dx.doi.org/10.1016/j.infsof.2008.01.006
Dyer, L., & Shafer, R. A. (2003). Dynamic organizations: Achieving marketplace and organizational agility with people.
Economist Intelligence Unit. (2009). *Organisational agility: How business can survive and thrive in turbulent times*.
Edmondson, A. C., & Nembhard, I. M. (2009). Product development and learning in project teams: The challenges are the benefits. *Journal of product innovation management, 26*(2), 123-138.
Eklund, U., & Bosch, J. (2012). Applying Agile Development in Mass-Produced Embedded Systems [Eklund2012]. In C. Wohlin (Ed.), *Agile Processes in Software Engineering and Extreme Programming: 13th International Conference, XP 2012, Malmö, Sweden, May 21-25, 2012. Proceedings* (pp. 31-46). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-30350-0_3. doi:10.1007/978-3-642-30350-0_3
Eklund, U., Olsson, H. H., & Strøm, N. J. (2014). Industrial challenges of scaling agile in mass-produced embedded systems*Springer.* Symposium conducted at the meeting of the International Conference on Agile Software Development
Faraj, S., & Sproull, L. (2000). Coordinating expertise in software development teams. *Management science, 46*(12), 1554-1568.
Farcic, V., & Garcia, A. (2015). *Test-Driven Java Development*: Packt Publishing. Retrieved from https://books.google.co.nz/books?id=tRl1CgAAQBAJ
Fenton-O'Creevy, M. (1998). Employee involvement and the middle manager: evidence from a survey of organizations. *Journal of Organizational Behavior*, 67-84.
Fowler, F. J. (2009). *Survey Research Methods*: SAGE Publications. Retrieved from https://books.google.co.nz/books?id=2Enm9gWeH2IC
Fowler, M., Beck, K., Brant, J., Opdyke, W., & Roberts, D. (2012). *Refactoring: Improving the Design of Existing Code*: Pearson Education. Retrieved from https://books.google.co.nz/books?id=HmrDHwgkbPsC
Gaddis, S. E. (1998). How to design online surveys [Article]. *Training & Development, 52*(6), 67.
Gandossy, R. (2003). The need for speed. *Journal of Business Strategy, 24*(1), 29-33. doi:doi:10.1108/02756660310508245
Gänswein, W. (2011). *Effectiveness of Information Use for Strategic Decision Making*: Gabler Verlag. Retrieved from https://books.google.co.nz/books?id=6DNGt1eY2ugC
Garton, L., Haythornthwaite, C., & Wellman, B. (1997). Studying online social networks. *Journal of Computer-Mediated Communication, 3*(1), 0-0.
Gefen, D., Straub, D., & Boudreau, M.-C. (2000). Structural equation modeling and regression: Guidelines for research practice. *Communications of the association for information systems, 4*(1), 7.
Gefen, D., Straub, D. W., & Rigdon, E. E. (2011). An update and extension to SEM guidelines for admnistrative and social science research. *Management Information Systems Quarterly, 35*(2), iii-xiv.
Giesecke, J., & McNeil, B. (2004). Transitioning to the learning organization.
Goebel, C. J. (2009). How being agile changed our human resources policies*IEEE.* Symposium conducted at the meeting of the Agile Conference, 2009. AGILE'09.
Goldman, S. L. (1995). *Agile competitors and virtual organizations: strategies for enriching the customer*: Van Nostrand Reinhold Company.

Goldman, S. L., & Nagel, R. N. (1993). Management, technology and agility: the emergence of a new era in manufacturing. *International Journal of Technology Management, 8*(1-2), 18-38.

Goldman, S. L., Nagel, R. N., & Preiss, K. (1995). *Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer*: Van Nostrand Reinhold. Retrieved from https://books.google.co.nz/books?id=ZBNPAAAAMAAJ

Goodpasture, J. C. (2015). *Project Management the Agile Way, Second Edition: Making it Work in the Enterprise*: J ROSS PUB Incorporated. Retrieved from https://books.google.co.nz/books?id=nF7hCgAAQBAJ

Gotwon, G. G., & Ditomaso, N. (1992). PREDICTING CORPORATE PERFORMANCE FROM ORGANIZATIONAL CULTURE [Article]. *Journal of Management Studies, 29*(6), 783-798.

Goyal, R. C. (2010). *Research Methodology for Health Professionals*: Jaypee Brothers,Medical Publishers Pvt. Limited. Retrieved from https://books.google.co.nz/books?id=yNv5AwAAQBAJ

Griffin, B., & Hesketh, B. (2003). Adaptable behaviours for successful work and career adjustment. *Australian Journal of psychology, 55*(2), 65-73.

Grisham, P. S., & Perry, D. E. (2005). Customer relationships and extreme programming*ACM.* Symposium conducted at the meeting of the ACM SIGSOFT Software Engineering Notes

Guang-yong, H. (2011, 27-29 May 2011). Study and practice of import Scrum agile software development Symposium conducted at the meeting of the 2011 IEEE 3rd International Conference on Communication Software and Networks doi:10.1109/ICCSN.2011.6013698

Guba, E. G., & Lincoln, Y. S. (1994). Competing paradigms in qualitative research. *Handbook of qualitative research, 2*(163-194), 105.

Gunasekaran, A. (1998). Agile manufacturing: Enablers and an implementation framework. *International Journal of Production Research, 36*(5), 1223-1247. doi:10.1080/002075498193291

Gunasekaran, A. (1999). Agile manufacturing: a framework for research and development. *International journal of production economics, 62*(1), 87-105.

Gupta, B., Iyer, L. S., & Aronson, J. E. (2000). Knowledge management: practices and challenges. *Industrial Management & Data Systems, 100*(1), 17-21. doi:doi:10.1108/02635570010273018

Guzzo, R. A., & Dickson, M. W. (1996). Teams in organizations: Recent research on performance and effectiveness. *Annual review of psychology, 47*(1), 307-338.

Hair, J. F., Hult, G. T. M., Ringle, C., & Sarstedt, M. (2016). *A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM)*: SAGE Publications. Retrieved from https://books.google.co.nz/books?id=Xn-LCwAAQBAJ

Hair, J. F., Ringle, C. M., & Sarstedt, M. (2011). PLS-SEM: Indeed a silver bullet. *Journal of Marketing theory and Practice, 19*(2), 139-152.

Hanks, B., McDowell, C., Draper, D., & Krnjajic, M. (2004). Program quality with pair programming in CS! *ACM SIGCSE Bulletin, 36*(3), 176-180.

Hansen, M. T., & Baggesen, H. (2009, 24-28 Aug. 2009). From CMMI and Isolation to Scrum, Agile, Lean and Collaboration Symposium conducted at the meeting of the 2009 Agile Conference doi:10.1109/AGILE.2009.18

Haugset, B., & Hanssen, G. K. (2008, 4-8 Aug. 2008). Automated Acceptance Testing: A Literature Review and an Industrial Case Study Symposium conducted at the meeting of the Agile 2008 Conference doi:10.1109/Agile.2008.82

Haugset, B., & Stalhane, T. (2012, 4-7 Jan. 2012). Automated Acceptance Testing as an Agile Requirements Engineering Practice Symposium conducted at the meeting of the 2012 45th Hawaii International Conference on System Sciences doi:10.1109/HICSS.2012.127

Hazzan, O., & Dubinsky, Y. (2009). *Agile Software Engineering*: Springer London. Retrieved from https://books.google.co.nz/books?id=kNFqQW3uZucC

Hedberg, M. (2015). Competences in Agile Development: Exploring the social, functional and cognitive requirements of a systems developer.
Henderson-Sellers, B., & Serour, M. (2005). Creating a dual-agility method: The value of method engineering. *Journal of Database Management, 16*(4), 1.
Herbsleb, J. D., & Mockus, A. (2003). An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering, 29*(6), 481-494. doi:10.1109/TSE.2003.1205177
Highsmith, J. A. (2002). *Agile Software Development Ecosystems*: Addison-Wesley. Retrieved from https://books.google.co.nz/books?id=uE4FGFOHs2EC
Hoda, R., Noble, J., & Marshall, S. (2013). Self-Organizing Roles on Agile Software Development Teams. *IEEE Transactions on Software Engineering, 39*(3), 422-444. doi:10.1109/TSE.2012.30
Hoegl, M., & Proserpio, L. (2004). Team member proximity and teamwork in innovative projects. *Research policy, 33*(8), 1153-1165.
Hopp, W. J., & Spearman, M. L. (2011). *Factory physics: Foundations of Manufacturing Management*: Waveland Press.
Hopp, W. J., & Van Oyen, M. P. (2004). Agile workforce evaluation: a framework for cross-training and coordination. *IIE Transactions, 36*(10), 919-940.
House, R. j. (1991). The distribution and exercise of power in complex organizations: A MESO theory. *The Leadership Quarterly, 2*(1), 23-58. doi:https://doi.org/10.1016/1048-9843(91)90005-M
Hoyt, J., Huq, F., & Kreiser, P. (2007). Measuring organizational responsiveness: the development of a validated survey instrument. *Management Decision, 45*(10), 1573-1594. doi:doi:10.1108/00251740710837979
Hsu, J. S.-C., Chan, C.-L., Liu, J. Y.-C., & Chen, H.-G. (2008). The impacts of user review on software responsiveness: Moderating requirements uncertainty. *Information & Management, 45*(4), 203-210. doi:http://dx.doi.org/10.1016/j.im.2008.01.006
Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader)*: Pearson Education. Retrieved from https://books.google.co.nz/books?id=6ADDuzere-YC
Hunt, J. (2006). *Agile Software Construction*: Springer London. Retrieved from https://books.google.co.nz/books?id=P8sWIoUWrdIC
Huntley, C. L. (2003). Organizational learning in open-source software projects: an analysis of debugging data. *IEEE Transactions on Engineering Management, 50*(4), 485-493. doi:10.1109/TEM.2003.820136
Iivari, J., & Iivari, N. (2011). The relationship between organizational culture and the deployment of agile methods. *Information and Software Technology, 53*(5), 509-520. doi:http://dx.doi.org/10.1016/j.infsof.2010.10.008
Janićijević, N. (2013). The mutual impact of organizational culture and structure. *Economic annals, 58*(198), 35-60.
Janis, I. L., & Mann, L. (1977). *Decision making: A psychological analysis of conflict, choice, and commitment*: Free Press.
Janz, B. D. (1999). Self-directed teams in IS: correlates for improved systems development work outcomes. *Information & Management, 35*(3), 171-192.
Jarvis, C. B., MacKenzie, S. B., & Podsakoff, P. M. (2003). A critical review of construct indicators and measurement model misspecification in marketing and consumer research. *Journal of consumer research, 30*(2), 199-218.
Jegerski, J., & VanPatten, B. (2013). *Research Methods in Second Language Psycholinguistics*: Taylor & Francis. Retrieved from https://books.google.co.nz/books?id=jMdiAgAAQBAJ
Jerez-Gómez, P., Céspedes-Lorente, J., & Valle-Cabrera, R. (2005). Organizational learning capability: a proposal of measurement. *Journal of Business Research, 58*(6), 715-725. doi:http://dx.doi.org/10.1016/j.jbusres.2003.11.002

Johannesson, P., & Perjons, E. (2014). Research Paradigms [Johannesson2014]. In *An Introduction to Design Science* (pp. 167-179). Cham: Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-319-10632-8_12. doi:10.1007/978-3-319-10632-8_12

Johnson, B., & Christensen, L. (2010). *Educational Research: Quantitative, Qualitative, and Mixed Approaches*: SAGE Publications. Retrieved from https://books.google.co.nz/books?id=b2ujHWrRpVQC

Kaplan, B., & Maxwell, J. A. (2005). Qualitative research methods for evaluating computer information systems. In *Evaluating the organizational impact of healthcare information systems* (pp. 30-55): Springer.

Katumba, B., & Knauss, E. (2014). Agile Development in Automotive Software Development: Challenges and Opportunities [Katumba2014]. In A. Jedlitschka, P. Kuvaja, M. Kuhrmann, T. Männistö, J. Münch, & M. Raatikainen (Eds.), *Product-Focused Software Process Improvement: 15th International Conference, PROFES 2014, Helsinki, Finland, December 10-12, 2014. Proceedings* (pp. 33-47). Cham: Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-319-13835-0_3. doi:10.1007/978-3-319-13835-0_3

Kavitha, R., & Ahmed, M. I. (2011). A knowledge management framework for agile software development teams*IEEE.* Symposium conducted at the meeting of the Process Automation, Control and Computing (PACC), 2011 International Conference on

Kelly, A. (2008). *Changing Software Development: Learning to Become Agile*: Wiley. Retrieved from https://books.google.co.nz/books?id=-4Nji7jhLXcC

Kelly, a. (2015). *Xanpan: Team Centric Agile Software Development*: Ipicturebooks. Retrieved from https://books.google.co.nz/books?id=5oyzBgAAQBAJ

Kessler, E. H., & Chakrabarti, A. K. (1996). Innovation Speed: A Conceptual Model of Context, Antecedents, and Outcomes [research article](4), 1143. Retrieved from http://ezproxy.aut.ac.nz/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edsjsr&AN=edsjsr.259167&site=eds-live

Khan, O. M. A. (2016). *JavaScript for .NET Developers*: Packt Publishing. Retrieved from https://books.google.co.nz/books?id=JevUDQAAQBAJ

Kivunja, C., & Kuyini, A. B. (2017). Understanding and Applying Research Paradigms in Educational Contexts. *International Journal of Higher Education, 6*(5), 26.

Kohn, A. (1999). *Punished by rewards: The trouble with gold stars, incentive plans, A's, praise, and other bribes*: Houghton Mifflin Harcourt.

Koskela, J., & Abrahamsson, P. (2004). On-site customer in an XP project: Empirical results from a case study*Springer.* Symposium conducted at the meeting of the European Conference on Software Process Improvement

Kotzab, H., Seuring, S., Müller, M., & Reiner, G. (2006). *Research Methodologies in Supply Chain Management*: Physica-Verlag HD. Retrieved from https://books.google.co.nz/books?id=LnGJJbImqngC

Kusiak, A., & He, D. (1997). Design for an enterprise. In *Enterprise Engineering and Integration* (pp. 420-430): Springer.

Laanti, M. (2014). Characteristics and principles of scaled agile*Springer.* Symposium conducted at the meeting of the International Conference on Agile Software Development

Lagerberg, L., Skude, T., Emanuelsson, P., Sandahl, K., St, D., x00E, & hl. (2013, 10-11 Oct. 2013). The Impact of Agile Principles and Practices on Large-Scale Software Development Projects: A Multiple-Case Study of Two Projects at Ericsson Symposium conducted at the meeting of the 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement doi:10.1109/ESEM.2013.53

Lal, R. (2011). *Strategic factors in agile software development method adaptation: a study of market-driven organisations: a thesis presented in partial [fulfilment] of the requirements for the degree of Doctor of Philosophy in Information Technology at Massey University, Albany campus, New Zealand*. Massey University.

Lam, A. (2000). Tacit knowledge, organizational learning and societal institutions: An integrated framework. *Organization studies, 21*(3), 487-513.
Landaeta, R. E., Viscardi, S., & Tolk, A. (2011). Strategic management of scrum projects: An organizational learning perspective*IEEE.* Symposium conducted at the meeting of the Technology Management Conference (ITMC), 2011 IEEE International
Laranjeiro, N., & Vieira, M. (2013). Adapting Test-Driven Development to Build Robust Web Services. *Software Design and Development: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications*, 50.
Laukkanen, E., Paasivaara, M., & Arvonen, T. (2015, 3-7 Aug. 2015). Stakeholder Perceptions of the Adoption of Continuous Integration -- A Case Study Symposium conducted at the meeting of the 2015 Agile Conference doi:10.1109/Agile.2015.15
Law, A., & Learn, S. (2005). Waltzing with changes [agile software development]*IEEE.* Symposium conducted at the meeting of the Agile Conference, 2005. Proceedings
Lee, G., & Xia, W. (2010). Toward agile: an integrated analysis of quantitative and qualitative field data on software development agility. *Mis Quarterly, 34*(1), 87-114.
Leffingwell, D. (2010). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*: Pearson Education. Retrieved from https://books.google.co.nz/books?id=pTExbNmZwZUC
Lei, D., Hitt, M. A., & Bettis, R. (1996). Dynamic core competences through meta-learning and strategic context. *Journal of management, 22*(4), 549-569.
Lewis, W. E. (2016). *Software Testing and Continuous Quality Improvement, Third Edition*: CRC Press. Retrieved from https://books.google.co.nz/books?id=fgaBDd0TfT8C
Li, Y. (2012). Workforce agility metric in EIS*IEEE.* Symposium conducted at the meeting of the 2012 International Conference on Information Management, Innovation Management and Industrial Engineering
Lin, C.-T., Chiu, H., & Chu, P.-Y. (2006). Agility index in the supply chain. *International Journal of Production Economics, 100*(2), 285-299.
Lin, C.-T., Chiu, H., & Tseng, Y.-H. (2006). Agility evaluation using fuzzy logic. *International Journal of Production Economics, 101*(2), 353-368. doi:http://dx.doi.org/10.1016/j.ijpe.2005.01.011
Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., . . . Zelkowitz, M. (2002). Empirical findings in agile methods*Springer.* Symposium conducted at the meeting of the Conference on Extreme Programming and Agile Methods
Loewen, S., & Plonsky, L. (2015). *An A–Z of Applied Linguistics Research Methods*: Palgrave Macmillan. Retrieved from https://books.google.co.nz/books?id=UCYpCwAAQBAJ
Mahmoudsalehi, M., Moradkhannejad, R., & Safari, K. (2012). How knowledge management is affected by organizational structure. *The Learning Organization, 19*(6), 518-528. doi:doi:10.1108/09696471211266974
Marczak, S., Kwan, I., & Damian, D. (2009, 31-31 Aug. 2009). Investigating Collaboration Driven by Requirements in Cross-Functional Software Teams Symposium conducted at the meeting of the 2009 Collaboration and Intercultural Issues on Requirements: Communication, Understanding and Softskills doi:10.1109/CIRCUS.2009.2
Marquardt, M. J. (1996). *Building the learning organization*: McGraw-Hill Companies New York, NY.
Martin, A., Noble, J., & Biddle, R. (2003). Being Jane Malkovich: A look into the world of an XP customer. *Extreme Programming and Agile Processes in Software Engineering*, 1012-1012.
Maskell, B. (2001). The age of agile manufacturing. *Supply Chain Management: An International Journal, 6*(1), 5-11. doi:doi:10.1108/13598540110380868
Mason-Jones, R., Naylor, B., & Towill, D. R. (2000). Engineering the leagile supply chain. *International Journal of Agile Management Systems, 2*(1), 54-61.
Mathiassen, L., & Pries-Heje, J. (2006). Business agility and diffusion of information technology: Springer.

Maximini, D. (2015). *The Scrum Culture: Introducing Agile Methods in Organizations*: Springer International Publishing. Retrieved from https://books.google.co.nz/books?id=ShojBgAAQBAJ
McConnell, S. (2004). *Professional Software Development*. Boston: Addison-Wesley.
McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2002). The effects of pair programming on performance in an introductory programming course. *ACM SIGCSE Bulletin, 34*(1), 38-42.
McKenna, E. F. (2000). *Business Psychology and Organisational Behaviour: A Student's Handbook*: Psychology Press. Retrieved from https://books.google.co.nz/books?id=v1fqm6WlOcMC
Mehlsen, M. Y. (2009). Introduction to Structural Equation Modelling using SPSS and AMOS. *Psyke & Logos, 30*(2), 5.
Melnik, G., & Maurer, F. (2004). Direct verbal communication as a catalyst of agile knowledge sharing*IEEE.* Symposium conducted at the meeting of the Agile Development Conference, 2004
Miles, R. E., Snow, C. C., Meyer, A. D., & Coleman, H. J. (1978). Organizational strategy, structure, and process. *Academy of management review, 3*(3), 546-562.
Miller, D. (1981). Toward a new contingency approach: The search for organizational gestalts. *Journal of management studies, 18*(1), 1-26.
Misra, S. C., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software, 82*(11), 1869-1890. doi:http://dx.doi.org/10.1016/j.jss.2009.05.052
Moch, M. K., & Morse, E. V. (1977). Size, Centralization and Organizational Adoption of Innovations. *American Sociological Review, 42*(5), 716-725. doi:10.2307/2094861
Moe, N. B., Dings, T., x0F, yr, Dyb, T., & x0E. (2008, 26-28 March 2008). Understanding Self-Organizing Teams in Agile Software Development Symposium conducted at the meeting of the 19th Australian Conference on Software Engineering (aswec 2008) doi:10.1109/ASWEC.2008.4483195
Mohammadi, S., Nikkhahan, B., & Sohrabi, S. (2008, 13-15 Oct. 2008). An Analytical Survey of "On-Site Customer" Practice in Extreme Programming Symposium conducted at the meeting of the International Symposium on Computer Science and its Applications doi:10.1109/CSA.2008.72
Monochristou, V., & Vlachopoulou, M. (2007). Requirements specification using user stories. *Agile Software Development Quality Assurance, 71*.
Moon, M. J. (1999). The Pursuit of Managerial Entrepreneurship: Does Organization Matter? *Public Administration Review, 59*(1), 31-43. doi:10.2307/977477
Moore, E., & Spens, J. (2008, 4-8 Aug. 2008). Scaling Agile: Finding your Agile Tribe Symposium conducted at the meeting of the Agile, 2008. AGILE '08. Conference doi:10.1109/Agile.2008.43
Moreira, M. E. (2013). *Being Agile: Your Roadmap to Successful Adoption of Agile*: Apress. Retrieved from https://books.google.co.nz/books?id=2ZErAQAAQBAJ
Muijs, D. (2010). *Doing Quantitative Research in Education with SPSS*: SAGE Publications. Retrieved from https://books.google.co.nz/books?id=apFMQHF768EC
Myers, M. D. (2013). *Qualitative Research in Business and Management*: SAGE Publications. Retrieved from https://books.google.co.nz/books?id=XZARAgAAQBAJ
Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Commun. ACM, 48*(5), 72-78. doi:10.1145/1060710.1060712
Netemeyer, R. G., Bearden, W. O., & Sharma, S. (2003). *Scaling Procedures: Issues and Applications*: SAGE Publications. Retrieved from https://books.google.co.nz/books?id=c2dN7HDbr7kC
Nidhra, S., Dondeti, J., Katikar, P., & Tekkali, S. (2012, 5-7 Sept. 2012). Implementing the concept of refactoring in software development Symposium conducted at the meeting

of the 2012 CSI Sixth International Conference on Software Engineering (CONSEG) doi:10.1109/CONSEG.2012.6349468
Nidumolu, S. (1995). The Effect of Coordination and Uncertainty on Software Project Performance: Residual Performance Risk as an Intervening Variable. *Information Systems Research, 6*(3), 191-219. doi:doi:10.1287/isre.6.3.191
Nimon, K. F. (2012). Statistical assumptions of substantive analyses across the general linear model: a mini-review. *Frontiers in psychology, 3*, 322.
Nonaka, I. (1994). A dynamic theory of organizational knowledge creation. *Organization science, 5*(1), 14-37.
Nosek, J. T. (1998). The case for collaborative programming. *Communications of the ACM, 41*(3), 105-108.
O'Reilly, C. A., Chatman, J., & Caldwell, D. F. (1991). People and organizational culture: A profile comparison approach to assessing person-organization fit. *Academy of management journal, 34*(3), 487-516.
Olsson, H., Sandberg, A., Bosch, J., & Alahyari, H. (2014). Scale and Responsiveness in Large-Scale Software Development. *IEEE Software, 31*(5), 87-93. doi:10.1109/MS.2013.139
Onoma, A. K., Tsai, W.-T., Poonawala, M., & Suganuma, H. (1998). Regression testing in an industrial environment. *Commun. ACM, 41*(5), 81-86. doi:10.1145/274946.274960
Passos, C., Mendon, M., & Cruzes, D. S. (2014, Sept. 28 2014-Oct. 3 2014). The Role of Organizational Culture in Software Development Practices: A Cross-Case Analysis of Four Software Companies Symposium conducted at the meeting of the Software Engineering (SBES), 2014 Brazilian Symposium on doi:10.1109/SBES.2014.12
Pérez-Bustamante, G. (1999). Knowledge management in agile innovative organisations. *Journal of knowledge management, 3*(1), 6-17.
Phalnikar, R., Deshpande, V., & Joshi, S. (2009). Applying agile principles for distributed software development*IEEE.* Symposium conducted at the meeting of the Advanced Computer Control, 2009. ICACC'09. International Conference on
Pichler, R. (2010). *Agile Product Management with Scrum: Creating Products that Customers Love (Adobe Reader)*: Pearson Education. Retrieved from https://books.google.co.nz/books?id=aLSu0P0EojEC
Pierce, J. L., & Delbecq, A. L. (1977). Organization structure, individual attitudes and innovation. *Academy of management review, 2*(1), 27-37.
Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of agile practices on communication in software development [journal article]. *Empirical Software Engineering, 13*(3), 303-337. doi:10.1007/s10664-008-9065-9
Plonka, F. E. (1997). Developing a lean and agile work force. *Human Factors and Ergonomics in Manufacturing & Service Industries, 7*(1), 11-20.
Pollard, N. (2016). *Getting Started with Agile Software Development*: BookRix. Retrieved from https://books.google.co.nz/books?id=dVCBCwAAQBAJ
Pring, R. (2000). The 'false dualism'of educational research. *Journal of Philosophy of Education, 34*(2), 247-260.
Project Management Institute. (2012). *Pulse of the Profession - Organizational Agility*: Project Management Institute.
Qin, R., & Nembhard, D. A. (2015). Workforce agility in operations management. *Surveys in Operations Research and Management Science, 20*(2), 55-69.
Quélin, B. (2000). Core competencies, R&D management and partnerships. *European Management Journal, 18*(5), 476-487. doi:http://dx.doi.org/10.1016/S0263-2373(00)00037-2
Quinn, R. E., & Rohrbaugh, J. (1983). A spatial model of effectiveness criteria: Towards a competing values approach to organizational analysis. *Management science, 29*(3), 363-377.
Ras, E., & Weber, S. (2009, 19-19 May 2009). Software organization platform: Integrating organizational and individual learning Symposium conducted at the meeting of the

Wikis for Software Engineering, 2009. WIKIS4SE '09. ICSE Workshop on doi:10.1109/WIKIS4SE.2009.5069997

Read, K., Melnik, G., & Maurer, F. (2005). Examining Usage Patterns of the FIT Acceptance Testing Framework [Read2005]. In H. Baumeister, M. Marchesi, & M. Holcombe (Eds.), *Extreme Programming and Agile Processes in Software Engineering: 6th International Conference, XP 2005, Sheffield, UK, June 18-23, 2005. Proceedings* (pp. 127-136). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/11499053_15. doi:10.1007/11499053_15

Reed, K., & Blunsdon, B. (1998). Organizational flexibility in Australia. *International Journal of Human Resource Management, 9*(3), 457-477.

Reifer, D. J., Maurer, F., & Erdogmus, H. (2003). Scaling agile methods. *IEEE Software, 20*(4), 12-14. doi:10.1109/MS.2003.1207448

Reis, H. T., & Judd, C. M. (2000). *Handbook of Research Methods in Social and Personality Psychology*: Cambridge University Press. Retrieved from https://books.google.co.nz/books?id=j7aawGLbtEoC

Resnick, S., Bjork, A., & de la Maza, M. (2011). *Professional Scrum with Team Foundation Server 2010*: Wiley. Retrieved from https://books.google.co.nz/books?id=YocPEY3YyoYC

Rico, D. F., Sayani, H. H., & Sone, S. (2009). *The Business Value of Agile Software Methods: Maximizing ROI with Just-in-time Processes and Documentation*: J. Ross Pub. Retrieved from https://books.google.co.nz/books?id=anLeaMMgmo0C

Rouse, W. B. (2007). Agile information systems for agile decision making. *Agile Information Systems*, 16.

Rubin, K. S. (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process*: Addison-Wesley. Retrieved from https://books.google.co.nz/books?id=HkXX65VCZU4C

Salo, O., & Abrahamsson, P. (2005). Integrating agile software development and software process improvement: a longitudinal case study*IEEE*. Symposium conducted at the meeting of the Empirical Software Engineering, 2005. 2005 International Symposium on

Sambamurthy, V., Bharadwaj, A., & Grover, V. (2003). Shaping agility through digital options: Reconceptualizing the role of information technology in contemporary firms. *MIS quarterly*, 237-263.

Sanchez, L. M., & Nagi, R. (2001). A review of agile manufacturing systems. *International Journal of Production Research, 39*(16), 3561-3600. doi:10.1080/00207540110068790

Sandberg, A. B., & Crnkovic, I. (2017, 20-28 May 2017). Meeting Industry-Academia Research Collaboration Challenges with Agile Methodologies Symposium conducted at the meeting of the 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP) doi:10.1109/ICSE-SEIP.2017.20

Santos-Vijande, M. L., López-Sánchez, J. Á., & Trespalacios, J. A. (2012). How organizational learning affects a firm's flexibility, competitive strategy, and performance. *Journal of Business Research, 65*(8), 1079-1089.

Santos, V. A., Goldman, A., Shinoda, A. C. M., & Fischer, A. L. (2011). A view towards Organizational Learning: An empirical study on Scrum implementation Symposium conducted at the meeting of the SEKE

Sarantakos, S. (2005). *Social Research*: Palgrave Macmillan. Retrieved from https://books.google.co.nz/books?id=yIjCQgAACAAJ

Savolainen, J., Kuusela, J., & Vilavaara, A. (2010). Transition to agile development-rediscovery of important requirements engineering practices*IEEE*. Symposium conducted at the meeting of the Requirements Engineering Conference (RE), 2010 18th IEEE International

Schein, E. H. (2010). *Organizational culture and leadership* (Vol. 2): John Wiley & Sons.

Schiel, J. (2009). *Enterprise-Scale Agile Software Development*: CRC Press. Retrieved from https://books.google.co.nz/books?id=qtwlk-AYYhYC
Schiel, J. (2016). *The ScrumMaster Study Guide*: CRC Press. Retrieved from https://books.google.co.nz/books?id=Be7RBQAAQBAJ
Schifferstein, H. N. J., & Hekkert, P. (2011). *Product Experience*: Elsevier Science. Retrieved from https://books.google.co.nz/books?id=iQnfJHjcVQ8C
Schmidt, C. (2015). *Agile Software Development Teams*: Springer International Publishing. Retrieved from https://books.google.co.nz/books?id=oVs-CwAAQBAJ
Schmidt, R., Lyytinen, K., & Mark Keil, P. C. (2001). Identifying software project risks: An international Delphi study. *Journal of management information systems, 17*(4), 5-36.
Schulz, M. (2001). The uncertain relevance of newness: Organizational learning and knowledge flows. *Academy of management journal, 44*(4), 661-681.
Schumacker, R. E., & Lomax, R. G. (2010). *A Beginner's Guide to Structural Equation Modeling*: Routledge. Retrieved from https://books.google.co.nz/books?id=58pWPxWPC90C
Schwaber, K., & Sutherland, J. (2012). *Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, And Leave Competitors In the Dust*: Wiley. Retrieved from https://books.google.co.nz/books?id=sdnAZOAuuDkC
Scotland, J. (2012). Exploring the philosophical underpinnings of research: Relating ontology and epistemology to the methodology and methods of the scientific, interpretive, and critical research paradigms. *English Language Teaching, 5*(9), 9.
Sedehi, H., & Martano, G. (2012, 17-19 Oct. 2012). Metrics to Evaluate & Monitor Agile Based Software Development Projects - A Fuzzy Logic Approach Symposium conducted at the meeting of the 2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement doi:10.1109/IWSM-MENSURA.2012.22
Selleri Silva, F., Soares, F. S. F., Peres, A. L., Azevedo, I. M. d., Vasconcelos, A. P. L. F., Kamei, F. K., & Meira, S. R. d. L. (2015). Using CMMI together with agile software development: A systematic review. *Information and Software Technology, 58*, 20-43. doi:http://dx.doi.org/10.1016/j.infsof.2014.09.012
Semeijn, J. H., Van Der Heijden, B. I. J. M., & Van Der Lee, A. (2014). Multisource Ratings Of Managerial Competencies And Their Predictive Value For Managerial And Organizational Effectiveness [Article]. *Human Resource Management, 53*(5), 773-794. doi:10.1002/hrm.21592
Shalloway, A., Beaver, G., & Trott, J. R. (2009). *Lean-Agile Software Development: Achieving Enterprise Agility*: Pearson Education. Retrieved from https://books.google.co.nz/books?id=8lm9_k_fvfkC
Shapiro, B. P. (2001). Sprint sell to close sales quickly: Boston: Harvard Business School Press.
Sharifi, H., & Zhang, Z. (1999). A methodology for achieving agility in manufacturing organisations: An introduction. *International journal of production economics, 62*(1), 7-22.
Shaye, S. D. (2008). Transitioning a team to agile test methods*IEEE.* Symposium conducted at the meeting of the Agile, 2008. AGILE'08. Conference
Sherehiy, B., Karwowski, W., & Layer, J. K. (2007). A review of enterprise agility: Concepts, frameworks, and attributes. *International Journal of industrial ergonomics, 37*(5), 445-460.
Shi, Z., Chen, L., & Chen, T.-e. (2011, 11-13 March 2011). Agile planning and development methods Symposium conducted at the meeting of the 2011 3rd International Conference on Computer Research and Development doi:10.1109/ICCRD.2011.5764064
Shinkle, C. M. (2009). Applying the Dreyfus model of skill acquisition to the adoption of Kanban systems at software engineering professionals (SEP)*IEEE.* Symposium conducted at the meeting of the Agile Conference, 2009. AGILE'09.

Shore, J., & Warden, S. (2008). *The Art of Agile Development*: O'Reilly Media, Incorporated. Retrieved from https://books.google.co.nz/books?id=2q6bAgAAQBAJ
Shrivastava, P. (1983). A TYPOLOGY OF ORGANIZATIONAL LEARNING SYSTEMS. *Journal of Management Studies, 20*(1), 7-28. doi:10.1111/j.1467-6486.1983.tb00195.x
Siggelkow, N., & Levinthal, D. A. (2005). Escaping real (non-benign) competency traps: linking the dynamics of organizational structure to the dynamics of search. *Strategic Organization, 3*(1), 85-115. doi:doi:10.1177/1476127005050521
Singh, K. (2007). *Quantitative Social Research Methods*: SAGE Publications. Retrieved from https://books.google.co.nz/books?id=-OMnt3CT-SwC
Sivanantham, V. (2012). Knowledge Management in Agile Projects. *Cognizant 20-20 Insights*.
Sliger, M., & Broderick, S. (2008). *The Software Project Manager's Bridge to Agility*: Pearson Education. Retrieved from https://books.google.co.nz/books?id=5Ir-Bgy95w4C
Sohaib, O., & Khan, K. (2010). Integrating usability engineering and agile software development: A literature review*IEEE.* Symposium conducted at the meeting of the Computer design and applications (ICCDA), 2010 international conference on
Sohrabi, R., Asari, M., & Hozoori, M. J. (2014). Relationship between Workforce Agility and Organizational Intelligence (Case Study: The Companies of" Iran High Council of Informatics"). *Asian Social Science, 10*(4), 279.
Solanki, P. (2009). *EARNED VALUE MANAGEMENT: Integrated View of Cost and Schedule Performance*: Global India Publications Pvt. Limited. Retrieved from https://books.google.co.nz/books?id=4lmXKNR46tIC
Srinivasan, J., & Lundqvist, K. (2009). Using agile methods in software product development: A case study*IEEE.* Symposium conducted at the meeting of the Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on
Stahl, D., Hallén, K., & Bosch, J. (2017). Continuous Integration and Delivery Traceability in Industry: Needs and Practices*IEEE.* Symposium conducted at the meeting of the Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on
Stamatis, D. H. (2002). *Six Sigma and Beyond: Statistics and Probability*: CRC Press. Retrieved from https://books.google.co.nz/books?id=iDnMBQAAQBAJ
Stamelos, I. G. (2007). *Agile Software Development Quality Assurance*: Information Science Reference. Retrieved from https://books.google.co.nz/books?id=TWscMx09ENkC
Stray, V. G., Moe, N. B., & Aurum, A. (2012, 5-8 Sept. 2012). Investigating Daily Team Meetings in Agile Software Projects Symposium conducted at the meeting of the 2012 38th Euromicro Conference on Software Engineering and Advanced Applications doi:10.1109/SEAA.2012.16
Sufian, Q., & Monideepa, T. (2013). Lean and agile supply chain strategies and supply chain responsiveness: the role of strategic supplier partnership and postponement. *Supply Chain Management: An International Journal, 18*(6), 571-582. doi:10.1108/SCM-01-2013-0015
Sultana, S., Motla, Y. H., Asghar, S., Jamal, M., & Azad, R. (2014, 26-28 Feb. 2014). A hybrid model by integrating agile practices for Pakistani software industry Symposium conducted at the meeting of the 2014 International Conference on Electronics, Communications and Computers (CONIELECOMP) doi:10.1109/CONIELECOMP.2014.6808600
Sumukadas, N., & Sawhney, R. (2004). Workforce agility through employee involvement. *Iie Transactions, 36*(10), 1011-1021.
Sundararajan, S., Bhasi, M., & Vijayaraghavan, P. K. (2014). Case study on risk management practice in large offshore-outsourced Agile software projects. *IET Software, 8*(6), 245-257.
Talaulicar, T., Grundei, J., & Werder, A. v. (2005). Strategic decision making in start-ups: the effect of top management team organization and processes on speed and comprehensiveness. *Journal of Business Venturing, 20*(4), 519-541.

Talby, D., Hazzan, O., Dubinsky, Y., & Keren, A. (2006, 23-28 July 2006). Reflections on reflection in agile software development Symposium conducted at the meeting of the AGILE 2006 (AGILE'06) doi:10.1109/AGILE.2006.45

Tata, J., & Prasad, S. (2004). Team Self-management, Organizational Structure, and Judgments of Team Effectiveness. *Journal of Managerial Issues, 16*(2), 248-265.

Tavakoli, H. (2012). *A Dictionary of Research Methodology and Statistics in Applied Linguistics*: Rahnama. Retrieved from https://books.google.co.nz/books?id=-UcbDQAAQBAJ

Tinsley, H. E. A., & Brown, S. D. (2000). *Handbook of Applied Multivariate Statistics and Mathematical Modeling*: Elsevier Science. Retrieved from https://books.google.co.nz/books?id=IIbMnrgTpWMC

Tjosvold, D., & Tjosvold, M. (2015). *Building the Team Organization: How To Open Minds, Resolve Conflict, and Ensure Cooperation*: Palgrave Macmillan. Retrieved from https://books.google.co.nz/books?id=B2AMCgAAQBAJ

Tolfo, C., & Wazlawick, R. S. (2008). The influence of organizational culture on the adoption of extreme programming. *Journal of Systems and Software, 81*(11), 1955-1967. doi:http://dx.doi.org/10.1016/j.jss.2008.01.014

Tolfo, C., Wazlawick, R. S., Ferreira, M. G. G., & Forcellini, F. A. (2011). Agile methods and organizational culture: reflections about cultural levels. *Journal of Software Maintenance and Evolution: Research and Practice, 23*(6), 423-441. doi:10.1002/smr.483

Tonelli, A. O., Bermejo, P. H. S., Santos, M. A., Zambalde, A. L., Oliveira, M. S. d., & Antonialli, L. M. (2013, 7-10 Jan. 2013). Agile Practices to Accelerate the Delivery of Software: A Quantitative Study with Software Professionals Symposium conducted at the meeting of the System Sciences (HICSS), 2013 46th Hawaii International Conference on doi:10.1109/HICSS.2013.75

Turk, D., France, R., & Rumpe, B. (2014). Limitations of agile software processes. *arXiv preprint arXiv:1409.6600*.

Turley, R. T., & Bieman, J. M. (1995). Competencies of exceptional and nonexceptional software engineers. *Journal of Systems and Software, 28*(1), 19-38.

Unhelkar, B. (2016). *The Art of Agile Practice: A Composite Approach for Projects and Organizations*: CRC Press. Retrieved from https://books.google.co.nz/books?id=ZqnMBQAAQBAJ

Vaidya, A. (2014). Does dad know best, is it better to do less or just be safe? adapting scaling agile practices into the enterprise. *PNSQC. ORG*, 1-18.

Van Veelen, B., Storms, P., & van Aart, C. (2006). Effective and efficient coordination strategies for agile crisis response organizations. *Proceedings of ISCRAM 2006*.

Vanhanen, J., & Lassenius, C. L. (2007). Perceived effects of pair programming in an industrial context*IEEE.* Symposium conducted at the meeting of the Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on

Vasileva, A., & Schmedding, D. (2016, 6-9 Sept. 2016). How to Improve Code Quality by Measurement and Refactoring Symposium conducted at the meeting of the 2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC) doi:10.1109/QUATIC.2016.034

Vazquez-Bustelo, D., Avella, L., & Fernández, E. (2007). Agility drivers, enablers and outcomes: empirical test of an integrated agile manufacturing model. *International Journal of Operations & Production Management, 27*(12), 1303-1332.

Verma, C., & Amin, S. A. (2010, 6-8 Sept. 2010). Significance of Healthy Organizational Culture for Superior Risk Management During Software Development Symposium conducted at the meeting of the Developments in E-systems Engineering (DESE), 2010 doi:10.1109/DeSE.2010.37

Vernadat, F. (1999). Research agenda for agile manufacturing. *International Journal of Agile Management Systems, 1*(1), 37-40.

VersionOne. (2016). *Manage Your Mission-Critical VersionOne Integrations with ALM Connect*.

Vesey, J. T. (1992). Time-to-market: Put speed in product development. *Industrial Marketing Management, 21*(2), 151-158. doi:http://dx.doi.org/10.1016/0019-8501(92)90010-Q
Vinzi, V. E., Chin, W. W., Henseler, J., & Wang, H. (2010). Perspectives on partial least squares. In *Handbook of Partial Least Squares* (pp. 1-20): Springer.
Wadhwa, S., & Rao, K. (2003). Enterprise modeling of supply chains involving multiple entity flows: role of flexibility in enhancing lead time performance. *Studies in informatics and control, 12*(1), 5-20.
Walczak, S. (2005). Organizational knowledge management structure. *The Learning Organization, 12*(4), 330-339. doi:doi:10.1108/09696470510599118
Wallach, E. J. (1983). Individuals and organizations: The cultural match. *Training & Development Journal*.
Wang, M. L., Hsu, B. F., Chen, W. Y., & Lin, Y. Y. (2008, 27-31 July 2008). Structural characteristics, process and effectiveness of cross-functional teams consisted of specialists and technicians in the healthcare industry Symposium conducted at the meeting of the PICMET '08 - 2008 Portland International Conference on Management of Engineering & Technology doi:10.1109/PICMET.2008.4599870
Wang, Q., Pfahl, D., & Raffo, D. (2008). *Making Globally Distributed Software Development a Success Story: International Conference on Software Process, ICSP 2008 Leipzig, Germany, May 10-11, 2008, Proceedings*: Springer Berlin Heidelberg. Retrieved from https://books.google.co.nz/books?id=f7FtCQAAQBAJ
Wegner, D. M. (1987). Transactive memory: A contemporary analysis of the group mind. In *Theories of group behavior* (pp. 185-208): Springer.
Weick, K. E. K. E. (1969). *The social psychology of organizing*.
Wellman, B. (1997). An electronic group is virtually a social network. *Culture of the Internet, 4*, 179-205.
Wendorff, P. (2002). Organisational culture in agile software development*Springer.* Symposium conducted at the meeting of the International Conference on Product Focused Software Process Improvement
Werder, K., Zobel, B., & Maedche, A. (2016). PDISC–Towards a Method for Software Product DISCovery*Springer.* Symposium conducted at the meeting of the International Conference of Software Business
West, D., & Grant, T. (2010). Agile development: Mainstream adoption has changed agility.
Wiig, K. M. (1997). Knowledge management: an introduction and perspective. *journal of knowledge Management, 1*(1), 6-14.
Wilburn, A. J. (1984). *Practical Statistical Sampling for Auditors*: Taylor & Francis. Retrieved from https://books.google.co.nz/books?id=V0WETIr0KSEC
Williams, L., & Cockburn, A. (2003). Guest Editors' Introduction: Agile Software Development: It's about Feedback and Change. *Computer, 36*(6), 39-43.
Williams, L., McDowell, C., Nagappan, N., Fernald, J., & Werner, L. (2003). Building pair programming knowledge through a family of experiments*IEEE.* Symposium conducted at the meeting of the Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on
Wong, K. K.-K. (2013a). Partial least squares structural equation modeling (PLS-SEM) techniques using SmartPLS.
Wong, K. K.-K. (2013b). Partial least squares structural equation modeling (PLS-SEM) techniques using SmartPLS. *Marketing Bulletin, 24*(1), 1-32.
Wright, K. B. (2005). Researching Internet-based populations: Advantages and disadvantages of online survey research, online questionnaire authoring software packages, and web survey services. *Journal of Computer-Mediated Communication, 10*(3), 00-00.
Xue-Mei, L., Guochang, G., Yong-Po, L., & Ji, W. (2009, March 31 2009-April 2 2009). Research and Implementation of Knowledge Management Methods in Software Testing Process Symposium conducted at the meeting of the Computer Science and Information Engineering, 2009 WRI World Congress on doi:10.1109/CSIE.2009.360

Yauch, C. A. (2007). Team-based work and work system balance in the context of agile manufacturing. *Applied Ergonomics, 38*(1), 19-27.
Yeung, A. K. (1999). *Organizational Learning Capability*: Oxford University Press. Retrieved from https://books.google.co.nz/books?id=QcTnCwAAQBAJ
Youndt, M. A., Snell, S. A., Dean, J. W., & Lepak, D. P. (1996). Human resource management, manufacturing strategy, and firm performance. *Academy of management Journal, 39*(4), 836-866.
Yun, G., & Trumbo, C. (2006). Comparative response to a survey executed by post, e-mail, and web form. 2000.
Yusuf, Y. Y., Sarhadi, M., & Gunasekaran, A. (1999). Agile manufacturing:: The drivers, concepts and attributes. *International Journal of production economics, 62*(1), 33-43.
Zammuto, R. F., & O'Connor, E. J. (1992). Gaining advanced manufacturing technologies' benefits: The roles of organization design and culture. *Academy of Management Review, 17*(4), 701-728.
Zander, U., & Kogut, B. (1995). Knowledge and the speed of the transfer and imitation of organizational capabilities: An empirical test. *Organization science, 6*(1), 76-92.
Zeng, M., Wang, C., & Long, Q. y. (2009, 19-21 May 2009). Practices of Extreme Programming for ERP Based on Two-dimensional Dynamic Time Scheduling Interface Method Symposium conducted at the meeting of the 2009 WRI World Congress on Software Engineering doi:10.1109/WCSE.2009.186
Zhou, Y. (2009, 7-8 March 2009). UniX Process, Merging Unified Process and Extreme Programming to Benefit Software Development Practice Symposium conducted at the meeting of the 2009 First International Workshop on Education Technology and Computer Science doi:10.1109/ETCS.2009.690

Appendix A. Participant Information Sheet

Participant Information Sheet

Date Information Sheet Produced:

9 May 2017

Project Title

Agility framework for software development: an investigation into agility concepts with the software development industry.

An Invitation

My name is Kevin Kusuma. I am a master student in the Computer and Information Sciences School at the Auckland University of Technology under the supervision of Dr Ramesh Lal. I would like to invite you to participate in my research to investigate agility elements in software development. Your participation in this study is voluntary and will take approximately forty-five to fifty minutes of your time. You are not obliged to take part in this research if you do not feel like it. You may withdraw from this study at any time before completion. Once you have done the survey, the survey data cannot be withdrawn. This survey is anonymous and information regarding the participants will not be published or disclosed in the study outputs.

What is the purpose of this research?

The purpose of this research is to propose an agility framework for agile software development processes. This framework will identify factors that software engineering organizations and vendors ought to consider improving or changing in their development environment to successfully deliver software products. Hence, the framework will provide the software engineering community with knowledge and understanding on agility competencies. This research will be conducted by me (Kevin) in order to write up a thesis for Master of Computer and Information Sciences. The findings of this research are also expected to be presented in software development conferences and to be written up in a journal article.

How was I identified and why am I being invited to participate in this research?

You were identified as a potential participant because your company has had at least one year of agile software development experience. The researcher has compiled a list (with email contact, phone contact and postal address) of potential participants consisting of 140 New Zealand software vendors and business organizations that have in-house software development teams. Most of these organizations are known to be practising software development based on the agile approach for longer than one year (known through the New Zealand Agile Software Development conference and the Facebook page of agile community groups in New Zealand). An email will be sent to invite a senior member of the software engineering team to take part in the research. The contact details of potential participants are compiled through channels such as personal connection of both supervisor and researcher including web search.

The researcher will enhance this list by including potential participants from overseas, mostly from Australia.

How do I agree to participate in this research?

Your participation in this research is voluntary (it is your choice). Completion of the survey will indicate your consent to participate in this research. You are able to withdraw from the study at any time. However, once the findings have been produced, removal of your data may not be possible.

What will happen in this research?

This study needs your agreement in order to participate in the survey. You will be asked to answer short, anonymous questions. The questionnaire contains sets of questions related to your opinion about the agile elements in software development. It will take approximately forty-five to fifty minutes to complete the survey. You can ignore the questions that you feel uncomfortable to answer. If you do not wish to continue participating at a certain point, you are free to leave the survey. Based on your responses, the researcher will use the findings to produce a thesis and related conference papers and journal articles.

What are the discomforts and risks?

There are no significant discomforts or risks involved in taking part in completing the questionnaire. Your participation is entirely voluntary. You will not be asked questions relating to your culture, religion, values or beliefs. Your responses are entirely confidential and will be used for research purposes only. You or your organization's details will not be identified in the thesis or in any other writeup.

How will these discomforts and risks be alleviated?

If you feel any discomfort, you may withdraw your survey participation at any time

What are the benefits?

Your responses will help to investigate the agility elements in software development. The findings of this research will provide understanding on agility attributes that agile practitioners may want to focus

on and consider in order to achieve agility in software development which could possibly enrich your future experience.

How will my privacy be protected?

Your participation is completely anonymous and your personal information or identity will not be associated with your response. Once the research project is completed, all information will be stored in locked storage at an AUT office. Only the associated researchers have authority to access the data. After six years, the data will be destroyed. You will not be identified in any outputs of this research.

What are the costs of participating in this research?

There is no cost for you to participate in this research except for forty-five to fifty minutes of your time.

What opportunity do I have to consider this invitation?

You can decide to accept or decline this invitation after going through this information sheet. If you would like to make further inquiries, contact details about researcher and supervisor are provided below in the contact details section. Once again, you are free to withdraw from the survey at any time before completing all the questions.

Will I receive feedback on the results of this research?

I will provide a summary and copy of the final results of my thesis once completed.

What do I do if I have concerns about this research?

Any concerns regarding the nature of this project should be notified in the first instance to the Project Supervisor, Dr Ramesh Lal, ramesh.lal@aut.ac.nz, +64 9 921 9999 ext 6323

Concerns regarding the conduct of the research should be notified to the Executive Secretary of AUTEC, Kate O'Connor, *ethics@aut.ac.nz* , 921 9999 ext 6038.

Whom do I contact for further information about this research?

Researcher Contact Details:

Kevin Kusuma, *xhd4124@aut.ac.nz*

Project Supervisor Contact Details:

Dr Ramesh Lal, *ramesh.lal@aut.ac.nz*, +64 9 921 9999 ext 6323

Approved by the Auckland University of Technology Ethics Committee on 10 May 2017, AUTEC Reference number 16/144.

Appendix B. Questionnaire

You are warmly invited to participate in a survey.

Dear Sir/Madam,

My name is Kevin Kusuma. I am a master student in the Computer and Information Sciences School at the Auckland University of Technology (AUT) under the supervision of Dr Ramesh Lal.

As part of my Master's thesis, I am conducting a survey on agile software development to investigate the agility elements. Specifically, I am trying to create a framework that will identify factors that software organizations and vendors ought to consider improving or changing in their development environment to successfully develop and deliver software products.

I would be extremely grateful if you would kindly help me in my research by completing the attached questionnaire.

All responses will be kept strictly confidential. Participation in this survey is voluntary and anonymous. Please click on the 'next' button below to undertake the survey. The survey will take approximately forty-five to fifty minutes to complete.

If you are interested in the results of the study, please provide your email address below.

Thank you very much for your valuable time. Your input in this research is highly appreciated.

Kevin Kusuma, Master Student
Auckland University of Technology (AUT)
55 Wellesley St E, Auckland, 1010
Auckland City, New Zealand
E-mail: kevinkusuma21@yahoo.com

Enter your email address:

_____

General Questions

1. What is the size of your software engineering team?
    - Less than 10
    - 10-50
    - Over 50

2. How long have you been working in an agile environment?
    - 1-2 years
    - 2-3 years
    - 3-5 years
    - 5-10 years
    - 10-15 years
    - >15 years

3. Which one of the following do you belong to?
    - In-house development team of a business organization/corporate/government department.
    - In-house development team of a software vendor.
    - Contracting IT/Software development team
    - Contractor

4. How many projects does your organization undertake in a year since agile adoption?
    - 1-3 Projects
    - 4-6 Projects
    - >6 Projects

5. What is the success rate of your projects after agile adoption?
    - < 30%
    - 30% - 50%
    - 51% - 75%
    - 76% - 90%
    - 91% - 99%
    - 100%

6. **Respondent can choose more than one practice.**
   Which of the following agile practices does your software engineering team use?
    - Product backlog
    - Sprint backlog
    - Daily stand-up meeting
    - Vision planning
    - Release planning
    - Sprint planning
    - Coding standard
    - Collective code ownership
    - Acceptance testing
    - Regression testing
    - Unit testing
    - Continuous integration
    - Common workspace
    - Burn-down/Burn-up chart
    - Pair programming
    - Refactoring

- Retrospective
- Test-driven development
- User story mapping
- Self-organizing teams

7. Which of the following methods are used by your organization?
   a) Agile method
   b) Hybrid method (practice from two different agile methods)
   c) Agile maturity framework
   If your answer is a, go to question 8.
   If your answer is b, go to question 9.
   If your answer is c, go to question 10.

8. Which agile method has your organization adopted?
   - Extreme programming (XP)
   - Scrum
   - Adaptive software development (ASD)
   - Dynamic systems development method (DSDM)
   - Feature-driven development (FDD)
   - Rapid application development (RAD)
   - Lean software development
   - Kanban
   - Agile modelling
   - Crystal clear methods
   - User-centered Design (UCD)

Go to Question 12.

9. **Respondent can choose more than one method.**
   Which methods make up your hybrid agile approach?
   - Extreme programming (XP)
   - Scrum
   - Adaptive software development (ASD)
   - Dynamic systems development method (DSDM)
   - Feature-driven development (FDD)
   - Rapid application development (RAD)
   - Lean software development
   - Kanban
   - Agile modelling
   - Crystal clear methods
   - User-centered Design (UCD)

Go to Question 12.

10. Which agile maturity framework does your company/organization use?
    - Disciplined agile delivery (DAD)
    - Scaled agile framework (SAFE)
    - Scrum of scrums
    - Internally created methods
    - Lean management
    - Agile portfolio management (APM)
    - Large-scale scrum (LESS)

11. **Respondent can choose more than one method.**
    What agile methods are part of your maturity framework?
    - Extreme programming (XP)
    - Scrum
    - Adaptive software development (ASD)
    - Dynamic systems development method (DSDM)
    - Feature-driven development (FDD)
    - Rapid application development (RAD)
    - Lean software development
    - Kanban
    - Agile modelling
    - Crystal clear methods
    - User-centered Design (UCD)

Team Effort

12. **Respondent can choose more than one answer**
    Who (role) in your organization initially compiles and is responsible for creating vision and roadmap plans?
    - Product manager
    - Business analyst
    - Marketing manager
    - Project manager
    - Product development manager
    - Software engineering manager
    - Functional managers
13. Does your organization have an independent body (steering committee/product planning team) for approving new features and product for development?
    - Yes
    - No
14. In the (your) organization, the steering committee/product planning team makes well thought-out and first-time-right development decisions.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
15. Product manager's collaboration with the (your) development team for input and feedback is critical to make effective vision/roadmap decisions.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
16. In the (your) development team, a cross-functional effort is required to create a reliable product backlog.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree

- Neutral
- Somewhat Agree
- Agree
- Strongly Agree

17. Reviewing product backlog more than once during the project timeline with the key stakeholders enables the (your) development team to deliver strategic benefits.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
18. Face-to-face communication is vital to dive deep for crucial information helping the (your) development team to create reliable product backlog with independent tasks.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
19. In the (your) development team, ability to make reliable task estimation is dependent upon acquiring solid understanding and knowledge of tasks.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
20. The (your) development team has clearly defined criteria for planning (creating) product backlogs.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
21. In the (your) development team, the on-site customer facilitates a better understanding of the project requirements.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
22. In the (your) development team, the on-site customer enables swift backlog planning.
    - Strongly Disagree
    - Disagree

- Somewhat Disagree
- Neutral
- Somewhat Agree
- Agree
- Strongly Agree

23. Product backlog allows the (your) development team to incorporate requests for change during the project timeline.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
24. The (your) whole team participates to achieve reliable sprint plans.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
25. Trust in the (your) development team is a key factor for signing up for sprint commitments.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
26. What approach does (your) development team use to write code?
    - Solo programming
    - Pair programming

    If your answer to question 26 is solo programming, go to question 27.
    If your answer to question 26 is pair programming, go to question 28.
27. The (your) development team uses code review practice to deliver high-quality code.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
28. The (your) development team achieves better quality features through pair programming.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree

- Strongly Agree

29. One to one meetings bring better project outcomes in (your) development team.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
30. Daily stand-up meetings encourage open communication in (your) development team.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
31. Daily stand-up meetings help the (your) development team to identify any problems with the tasks being undertaken.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
32. Daily stand-up meetings enable the (your) development team to effectively coordinate sprint/iteration tasks in projects.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
33. Daily stand-up meetings enable the (your) development team to identify how the team is going to be productive for the day.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree

Competencies

34. Continuous integration practice is one of the critical practices for the (your) development team.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral

- Somewhat Agree
- Agree
- Strongly Agree

35. Continuous integration enables reliability of software releases by the (your) development team.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
36. Continuous integration enables (your) development team to be more productive.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
37. Continuous integration allows the (your) development team to fix most of the bugs before release.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
38. The (your) development team consists of individuals able to perform formal and informal (i.e. multiple) roles required in the project.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
39. The (your) development team consists of individuals with substantial local business, customer/client and product knowledge.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
40. The (your) development team consists of individuals with substantial local technical product knowledge.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree

- Neutral
- Somewhat Agree
- Agree
- Strongly Agree

41. The (your) development team consists of a good number of senior individuals.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
42. The (your) development team has a fully integrated test-driven development environment.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
43. The (your) development team has the capability to consistently deliver working software in short development cycles.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
44. Test-driven development has enabled the (your) development team to deliver working software on a regular basis.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
45. The (your) development team is fully capable of implementing unit tests.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree

Responsiveness

46. The empowerment of the (your) development team is a critical factor for the ability to learn and quickly adapt.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
47. The (your) development team has the ability to accept change in requirements during projects.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
48. The (your) development team has on the fly adopted or has the ability to adopt new development practices and skills.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
49. The (your) development team does beta releases of features to get useful insights and feedback to deliver useful features for the marketplace.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree

Organizational Learning

50. The (your) organizational product planning process (at a business level) has well thought-out practices and structures.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
51. The (your) software engineering level has well thought-out practices and structures (functional unit and roles) based on continuous reviews and reflections.
    - Strongly Disagree
    - Disagree

- Somewhat Disagree
- Neutral
- Somewhat Agree
- Agree
- Strongly Agree

52. The learning and understanding on the organizational core competency has enabled the (your) development team to identify and learn firm-specific development competencies.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
53. The team-effort, ownership and cross-functional cooperation in the (your) development team are enforced through adoption of practices mutually accepted by all the stakeholders.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
54. The (your) development team is also driven by the mind-set for continuous learning in projects.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
55. Task sharing in the (your) development team helps to gain in-depth understanding on other roles, responsibilities and skills.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
56. The (your) development team has the ability to solve unfamiliar problems and is able to cope with stressful situations in the projects.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree

Organizational Culture

57. The (your) organization has a supportive mind-set based on cross-functional effort.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
58. Face-to-face interaction and spontaneous collaboration are a critical culture in the (your) development team/organization.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
59. Individuals in the (your) team take part in a wide variety of work and decision-making.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
60. Collective decision-making is another critical element in the (your) development team/organization.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
61. Retrospective and continuous feedback of improvements is a critical practice in the (your) development team/organization.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
62. Collective responsibility is a key part of the (your) development team's work practices.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree

Workforce Agility

63. The (your) development team consists of individuals with appropriate skills and knowledge to carry out multiple tasks.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
64. The (your) development team consists of highly skilled and competent individuals.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
65. The (your) development team has negotiation capabilities and consensus behaviour ability to accept change, generate new ideas and accept new responsibilities.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree

Speed

66. The (your) development team only works the normal hours per day or week on a consistent basis to deliver projects.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
67. Development infrastructure of the (your) development team does not place limitations to team's expected work pace (speed).
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
68. The (your) development team achieves reliable product backlogs to ensure the expected speed for implementation.
    - Strongly Disagree
    - Disagree

- Somewhat Disagree
- Neutral
- Somewhat Agree
- Agree
- Strongly Agree

69. To ensure a consistent work pace (speed) in projects, the (your) development team makes certain that the team members have a solid understanding of the product backlog.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
70. The work practice of the (your) development team to create a reliable sprint plan requires a collective effort involving all the software engineers.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
71. Avoiding technical debt is a critical mind set of (your) development team even if it affects the delivery speed of your team.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
72. TDD (Test-Driven Development) and refactoring are essential work practices to delivery software in short development cycles even though it affects the delivery speed of the (your) development team.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree

Knowledge Management

73. In the (your) development team, significant local business and technical knowledge plays a vital role to make swift and first-time-right decisions (planning, design, implementation).
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree

- Agree
- Strongly Agree

74. Collective effort provides solid critical thinking and problem solving ability in the (your) development team.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
75. In the (your) development team, task sharing in projects is vital to build knowledge and experience.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree
76. In the (your) development team, work (method) practices also build experience and knowledge for undertaking future projects.
    - Strongly Disagree
    - Disagree
    - Somewhat Disagree
    - Neutral
    - Somewhat Agree
    - Agree
    - Strongly Agree