CRITICAL SUCCESS FACTORS FOR LARGE AND DISTRIBUTED AGILE SOFTWARE DEVELOPMENT PROJECTS USING SCRUM IN U.S.-BASED GLOBAL COMPANIES

by

Lorena Stanberry

MARCOS J. MONTERO-LARES, DPA, Faculty Mentor and Chair

BRUCE LAVIOLETTE, PhD, Committee Member

WILLIAM J. McKIBBIN, PhD, Committee Member

Rhonda Capron, EdD, Dean, School of Business and Technology

A Dissertation Presented in Partial Fulfillment

Of the Requirements for the Degree

Doctor of Business Administration

Capella University

February 2018

Abstract

This study expands upon research previously conducted on critical success factors for the implementation of agile software development methodologies. The purpose was to examine the relationships between 12 independent variables, representing possible critical success factors for agile software development projects (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule); and the dependent variable of project success, consisting of four dimensions (Quality, Scope, Time, and Cost). Participants in the study included 132 practitioners in U.S.-based global companies that have served as product owner, Scrum master, software developer, business analyst, and/or tester, for a completed large and distributed agile software development project using Scrum methodology. Graphical and quantitative data analysis techniques served to examine the study research model and test the hypotheses. Findings from data analysis support that all 12 critical success factors have an impact on the successful resolution of agile software development projects using Scrum methodology in U.S.-based global companies; however, with differing levels of significance. The results reflect all 12 factors are not significant for one or more of four dimensions of project success. Also, results support that five of the 12 critical success factors are significant; however, of these, three ranked higher than the others, and showed a significant effect on more than one of the dimensions of project success. These three factors are Delivery Strategy, Team Capability, and Project Definition Process.

Dedication

This dissertation is dedicated to my girls, Shawna and Heather, who never stop supporting and believing in me. It is also dedicated to my friends and peers Kimberly, John, and Jim, who struggled with and supported me as we all achieved our goals and milestones throughout the process.

Acknowledgments

I would like to thank my mentor, Dr. Marcos Montero-Lares for his guidance, support and advice throughout the entire process. I would also like to thank my committee members Dr. Bruce Laviolette and Dr. William McKibbin for their valuable feedback and guidance. My entire committee was instrumental in helping ensure quality research which contributes to the field of project management.

Table of Contents

List of Tables

List of Figures

CHAPTER 1. INTRODUCTION

Introduction

Companies of all sizes have used project management for years to coordinate change efforts, both large and small (Project Management Institute [PMI], 2017). Although the traditional project management (TPM) approach has been commonplace, the advancement and growth of technology and the global expansion of organizations has brought the emergence of new and more flexible approaches (Fernandez & Fernandez, 2008; Saynisch, 2010a). Over the last two decades, "the software development industry has seen the emergence of agile methodologies" (Matalonga, Solari, & Matturro, 2013, p. 1290) as an alternative to TPM methods. Matalonga et al. (2013) highlighted that the most important proposition of the agile movement, as outlined in the *Agile Manifesto* ("Manifesto for Agile Software Development," 2001), is "the belief in … the software developer, high feedback, close customer collaboration and just enough planning and documentation" (p. 1290). Once agile methods hit the market, adoption quickly occurred for software development using methodologies such as Scrum, Extreme Programming (XP), Adaptive Software Development (ASD), and Feature-Driven Development (Jeremiah, 2015). Although the use of agile methodologies has resulted in more successful projects, there are still opportunities to improving their rate of success in large software development projects (Hastie & Wojewoda, 2015; Hoda & Murugesan, 2016; VersionOne.com, 2016a).

By using an explanatory, quantitative, and survey research design, this doctoral study contributes to the existing body of knowledge in the field of Project Management by examining the significance of 12 possible factors identified by Chow and Cao (2008) as contributors to the success of large software development projects that use Scrum methodology in U.S.-based global companies. This chapter serves as an introduction to the study and addresses the background, business problem, research purpose, research questions, rationale, theoretical framework, and significance, the definition of terms, assumptions, and limitations.

Background

The practice of project management has grown and evolved throughout the years. Since its creation in 1969, the Project Management Institute (PMI) has contributed to the adoption of TPM methodologies, strategies, and best practices worldwide. The PMI (2013) defined TPM as the application of skills, knowledge, and experience to meet project requirements. TPM involves the completion of five phases (or process groups) under the guidance of a project manager, including initiation, planning, executing, monitoring and controlling, and closing. TPM also follows an iterative and progressively elaborated process commonly used for repetitive type projects (Fernandez & Fernandez, 2008; PMI, 2013).

As the practice of project management has grown and matured, organizations and practitioners have found that TPM methods do not deliver the results needed and expected for completeness and timeliness (Fernandez & Fernandez, 2008; Saynisch, 2010a). Similarly, scholars have found that TPM proposes a linear structure that lacks the flexibility to adjust quickly to challenges, opportunities, and changes. Therefore, it has been argued that TPM is not

fast moving enough to address the complexity of modern software development projects (Fernandez & Fernandez, 2008; Laanti & Abrahamsson, 2011; Saynisch, 2010b).

Advances in technology have brought about an increasing need for flexibility, responsiveness, efficiency, and effectiveness, which is why agile methods have emerged as a new development process (Chow & Cao, 2008). The *Agile Manifesto* set the foundation for the agile movement by providing an official definition and 12 principles to guide agile software development. As the *Agile Manifesto* noted:

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work, we have come to value: Individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, [and] responding to change over following a plan. That is, while there is value in the items on the right, we value the items on the left more. (para. 1)

Agile Alliance (2016) defined agile software development/engineering as an overarching term for a group of methods that are rooted in the concepts found in the *Agile Manifesto*. These methods have a foundation on self-organizing, cross-functional team collaboration. Agile software development entails an iterative process that allows for responsiveness and adaptability (Cao, Mohan, Xu, & Ramesh, 2009; Fernandez & Fernandez, 2008). Agile software development methodologies have cycles that are repetitive throughout the process along with constant feedback loops that receive and address customer feedback (Fernandez & Fernandez, 2008).

Agile software development methodologies include Scrum, Lean, Kanban, XP, ASD, Crystal, Dynamic Systems Development Method (DSDM), and Feature-Drive Development (FDD) (VersionOne, 2016b). Although these are different variations of agile software development methodologies/frameworks, they all share the same philosophy, along with many

characteristics and practices (VersionOne, 2016b). Nevertheless, agile software development methodologies differ in their use of terminology and approach for implementing agile principles (McLaughlin, 2016; VersionOne, 2016b).

Agile software development methodologies have continued to grow in popularity over the last decade (Dyba & Dingsoyr, 2008; Jeremiah, 2015; VersionOne, 2016a). For instance, VersionOne (2016a) asserted that the popularity of agile software development methods is partially due to their iterative style and attentiveness to improving team collaboration, quality, and the overall satisfaction of customers. Research conducted by Dyba and Dingsoyr (2008) showed an increasing interest in agile software development by companies, with 14% already applying it and 49% interested in adopting it. In a survey conducted by Jeremiah (2015), two-thirds of organizations are either using agile or leaning towards using it. Moreover, among agile software development methodologies, the use of Scrum continues to grow (Dyba & Dingsoyr, 2008; Ghani, Bello, & Bagiwa, 2015; Papatheocharous & Andreou, 2014; Scrum Alliance, 2016b; VersionOne, 2016a). Surveys (Ghani et al., 2015; Papatheocharous & Andreou, 2014; Scrum Alliance, 2016b; VersionOne, 2016a) have also reported the use of agile software development by over 50% of respondents, with some showing that nearly 70% of respondents use Scrum.

Scrum is an agile method that allows teams to work through complex adaptive problems while also deliver the product of highest value (Scrum Alliance, 2016a). Jeff Sutherland first introduced Scrum methodology in 1993 by borrowing the term *scrum* from a 1986 *Harvard Business Review* article by Takeuchi and Nonaka. In that article, Takeuchi and Nonaka compared high-performing, cross-functional teams to the "scrum" formation used in rugby (Alliance, 2016b). In the Scrum methodology, the product owner collaborates with business

stakeholders to manage the scope and facilitate the prioritization of the product backlog (McLaughlin, 2016; VersionOne, 2016b). The product backlog guides the work of a cross-functional team. This team consists of developers, business analysts, and testers, which collaborate to organize work in time frames called "sprints" in order to deliver a potentially shippable set of software features in an expedited fashion (VersionOne, 2016b). Additionally, the Scrum master, a part of the software development team, manages the process by coaching members, coordinating and facilitating ceremonies, and creating artifacts (Scrum Alliance, 2016c).

Even though the Scrum methodology has grown in popularity, organizations and practitioners find the implementation and scaling up processes to be a challenge (Ambler, 2014a; Chikhale & Mansouri, 2015; Papatheocharous & Andreou, 2014). Scholars have identified common issues and challenges reported by organizations. These include, but are not limited to, the management and teams' use of the methodology along with the lack of executive sponsorship and planning (Cao et al., 2009; Fernandez & Fernandez, 2008; Hoda & Murugesan, 2016; Khalil & Khalil, 2016). Additional challenges include the geographical distribution and cross-functionality of teams, along with an overemphasis on quick results with minimal amounts of testing (Cao et al., 2009; Fernandez & Fernandez, 2008; Hoda & Murugesan, 2016; Khalil & Khalil, 2016).

Many scholars (e.g., Ambler, 2014a; Khalil & Khalil, 2016) have also pointed out some key challenges that organizations need to address when scaling agile software development methods. These organizational challenges include the: adoption of technical practices, the transformation of culture, and changes in information technology (IT). Also, practitioners using Scrum methodology have also reported challenges to team experience, and particularly,

difficulties for self-organizing, changing organizational culture and delivering products on time (Ghani et al., 2015; Hoda & Murugesan, 2016). Given that organizations continue to face myriad issues, barriers, and challenges when implementing and using agile software development methodologies, including Scrum, it is important to identify and understand factors that are critical to support successful implementation. The factors that need to be right for project success are also known as critical success factors, or CSFs (Bullen & Rockart, 1981; Gandomani, Zulzalil, Abdul Ghani, Md Sultan, & Sharif, 2014)

Scholars have applied CSF theory to the study of software development methods, including agile software development methodologies (e.g., Brown, 2015; Chow & Cao, 2008; Stankovic, Nikolic, Djordjevic, & Cao, 2013). Extant research (e.g., Brown, 2015; Chow & Cao, 2008; Matalonga et al., 2013; Misra, Kumar, & Kumar, 2009; Stankovic et al., 2013) has identified related CSFs for the implementation of agile software development methodologies. However, there is agreement on the need for further research. For instance, Chow and Cao (2008) conducted seminal research on CSFs that could aid the implementation process and improve the success rate of agile software development projects. Their research focused on 12 possible CSFs with a sample of 109 projects worldwide. Brown (2015) adapted Chow and Cao's research model by focusing on 127 U.S. individuals involved in software development projects. Both Brown (2015) and Chow and Cao (2008) have acknowledged the need for additional research and recommended replicating their studies with a focus on a specific agile software development methodology to see if the results would vary or produce new significant CSFs.

Business Problem

As U.S.-based global companies increase and scale their use of agile software development methodologies, the rate of success of their projects has been lower than expected (Brown, 2015; Hastie & Wojewoda, 2015; Senapathi & Srinivasan, 2012). Many scholars (e.g., Brown, 2015; Hastie & Wojewoda, 2015; Senapathi & Srinivasan, 2012) have advocated for an increased use of agile methodologies as they are useful for addressing changing regulatory climates, vague system requirements, miscommunication, and demands for quick turnaround times in software development projects. The 2015 Standish Group's CHAOS Report (as cited in Hastie & Wojewoda, 2015) suggested that although the use of agile software development methodologies has resulted in more successful projects, there are still opportunities for improvement. The report noted that of those software development projects categorized as large, 18% were successful, 59% faced significant challenges, and 23% failed.

Leaders of global companies are increasingly confronted with business decision-makers and project managers who are not capable of effectively using Scrum methodology in large agile software development projects with distributed project teams (Gandomani et al., 2014; Gonçalves & Lopes, 2014). Laanti and Abrahamsson (2011) and Papatheocharous and Andreou (2014) found that agile software development projects fail because project team members (e.g., developers, business analysts, testers, product owners, management) frequently lack the experience and training needed to deploy and scale agile software development processes.

Business decision-makers and project managers seek out advice regarding the selection of factors needed to increase the likelihood of success in large and distributed agile software development projects (Papatheocharous and Andreou, 2014). Accordingly, scholars have recommended additional research on CSFs for agile software development projects to address

the distinct features of specific methodologies such as Scrum and their implementation in global companies (Brown, 2015) and distributed teams (Matalonga et al., 2013). Research on the significance of CSFs in large and distributed agile software development projects that use Scrum methodology is valuable (Brown, 2015; Gonçalves & Lopes, 2014; Matalonga et al., 2013). Such research, especially that on challenges specific to U.S.-based global companies, adds to the existing body of knowledge in the field of project management.

Research Purpose

The purpose of this study was to examine the relationships between 12 independent variables (representing possible CSFs for agile software development projects) and the dependent variable of project success (consisting of four dimensions), as proposed by Chow and Cao (2008) and later adapted by Brown (2015). The 12 independent variables include: (a) management commitment, (b) organizational environment, (c) team environment, (d) team capability, (e) customer involvement, (f) project management process, (g) project definition process, (h) agile software techniques, (i) delivery strategy, (j) project nature, (k) project type, and (l) project schedule. The four dimensions of project success include (a) quality, (b) scope; (c) time, and (d) cost. A web-based survey developed by Chow and Cao that consists of questions with a seven-point Likert scale served to measure these dependent and independent variables.

This study contributes to the existing body of knowledge in the field of project management by furthering understanding about factors to consider for the successful implementation of the Scrum methodology for large and distributed agile software development projects in U.S.-based global companies. Scrum has become the most popular agile software

development methodology in the past few years. Both scholars and practitioners (e.g., Dyba & Dingsoyr, 2008; Ghani et al., 2015; Papatheocharous & Andreou, 2014; VersionOne, 2016a) have recommended further research on CSFs for the successful implementation of Scrum methodology in large and distributed agile software development projects. This study expands on research conducted by Chow and Cao (2008) and Brown (2015) by concentrating on Scrum methodology in order to better understand the CSFs for its implementation in large and distributed software development projects.

Research Questions

This doctoral study examines the significance of CSFs for large and distributed agile software development projects that use Scrum methodology in U.S.-based global companies. The study expanded upon Chow and Cao's (2008) research examining the significance of 12 identified factors (representing five categories – Organizational, People, Process, Technical, and Project) as contributors to the success (in each of four dimensions – Quality, Scope; Time, and Cost) of large and distributed agile software development projects using Scrum methodology in U.S.-based global companies. Accordingly, the research questions that guided this study are as follows:

RQ1: To what extent do Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) predict the quality of agile software development projects?

RQ2: To what extent do Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) predict the scope of agile software development projects?

RQ3: To what extent do Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) predict the time of agile software development projects?

RQ4: To what extent do Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) predict the cost of agile software development projects?

Rationale

Seminal work on CSFs goes back to Daniel (1961), who noted that managers receive many different types of information, but not always the needed insights to make decisions, set strategies, and/ or measure goals accomplished. Rockart (1979) introduced the CSFs theory suggesting that rather than managing everything, managers would be more successful if they could identify and monitor a few critical indicators of the company's health. Bullen and Rockart (1981) further defined CSFs as the few key areas that must be done correctly to achieve success.

Scholars (e.g., Brown, 2015; Chow & Cao, 2008; Hoda & Murugesan, 2016; Laanti & Abrahamsson, 2011; Matalonga et al., 2013; Misra et al., 2009; Stankovic et al., 2013) have applied the CSF theory to project management and technology. These same scholars have identified possible CSFs for the implementation of agile software development methodologies, suggesting the need for further research.

Chow and Cao (2008) adopted an approach to research identified a set of 12 possible CSFs that could be applied to project success categories – Quality, Scope, Time, and Cost. Use of their approach occurred in various subsequent studies (e.g., Brown, 2015; Stankovic et al., 2013). Chow and Cao's set of 12 possible CSFs for agile software development projects includes: (a) management commitment, (b) organizational environment, (c) team environment, (d) team capability, (e) customer involvement, (f) project management process, (g) project definition process, (h) agile software engineering techniques, (i) delivery strategy, (j) project nature, (k) project type, and (l) project schedule. After testing 48 hypotheses, Chow and Cao (2008) pointed out that three of these CSFs – delivery strategy, agile software engineering techniques, and team capability – are significant.

Brown (2015) built on Chow and Cao's research by arguing for understanding CSFs within the context of a particular agile methodology. Khalil and Khalil (2016) identified the need for organizations to make decisions regarding "agile principles, agile architecture, agile infrastructure, project team needs, and agile project investment and prioritization" (p. 114).

By building upon and replicating the study by Chow and Cao (2008), this dissertation focuses on the CSFs for large and distributed agile software development projects using Scrum methodology in U.S.-based global companies. Chow and Cao's 2008 study brings beneficial information to the users of agile software development methodologies; however, there are

significant limitations to their study, including the small representation of both agile software development methods and U.S.-based projects. In his adaptation of Chow and Cao's research model, Brown (2015) also noted research limitations with a bias towards the XP and Feature-Driven Development methodologies. Chow and Cao and Brown recommended further replication of their studies within the context of specific agile software development methods and over a longer period of maturation and exposure to these methodologies. Chow and Cao specifically recommended repeating their study in five-to-ten years to see if new CSFs emerge or if some are no longer critical.

Finally, the growth of Scrum as one of the most popular agile software development methodologies supports the need for advancing further research on its implementation and scaling (Ghani et al., 2015; Papatheocharous & Andreou, 2014; Scrum Alliance, 2016c; VersionOne, 2016a). Chow and Cao (2008) and Brown (2015) recognized the value of research on CSFs for the implementation of Scrum methodology in large and distributed agile software development projects, and they further recommended future research on specific agile software development methodologies.

Theoretical Framework

This doctoral study applied a post-positivist, quantitative, and empirical approach (Creswell, 2014) to the examination of the significance of CSFs for large and distributed agile software development projects using Scrum methodology in U.S.-based global companies. An explanatory, quantitative, and survey research design served for examining the research model and testing the hypotheses regarding CSFs for the implementation of scrum agile software development methodologies.

Chow and Cao (2008) used a research model based on the concept, theory, and method of CSF (Bullen & Rockart, 1981; Rockart, 1979; Rockart & Crescenzi, 1984), which identifies and applies those "limited number of areas in which satisfactory results will ensure successful competitive performance. CSFs are the few key areas where 'things must go right' for the business to flourish and for the managers to attain their goal" (Bullen & Rockart, 1981, p. 7). CSF theory applies to this study because, as Daniel (1961) and Rockart (1979) noted, the CSF approach helps management to focus on the areas of most importance for successful implementation, adoption and scaling of agile software development methodologies.

As illustrated in Figure 1, Chow and Cao's (2008) research model reflects the 12 potential factors within five categories, including organizational factors people factors, process factors, technical factors, and project factors. Chow and Cao's (2008) research model served as the foundation for this study and provided variables, hypothesized relationships, and a survey instrument for examining the significance of CSFs for large and distributed agile software development projects using Scrum methodology in U.S.-based global companies. By focusing on Scrum methodology and global agile software development projects, this study contributes to the body of knowledge in the field of project management by identifying CSFs that allow for improved ease and minimized risk in the transition and adoption of Scrum methodology.

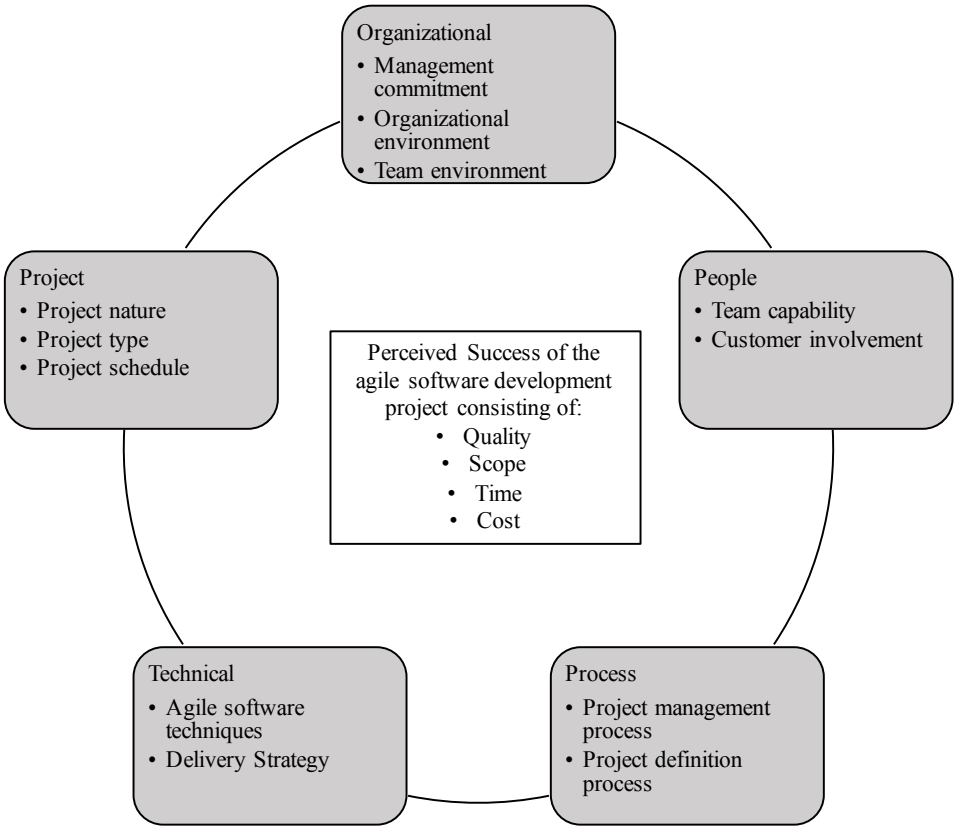

Figure 1. CSFs in Agile Software Projects. The figure shows the five groups with 12 CSFs that may impact the success of large and distributed agile software development projects using Scrum methodology.

Significance

The study has implications for both scholarship and business practice.

Implications for Scholarship

The study contributes to the scholarship in the field of project management by furthering research on CSFs for large and distributed agile software development projects using Scrum methodology in U.S.-based global companies and addressing limitations and recommendations noted in previous studies (e.g., Brown, 2015; Chow & Cao, 2008). Chow and Cao (2008) and Brown (2015) researched CSFs for agile software development projects with results reflecting heavily on participants using the XP methodology. Chow and Cao (2008) recommended three

areas for further research. These include (a) furthering representation of other agile software development methodologies, (b) using a larger sample of participants for strengthening statistical analysis, and (c) adjusting the sampling frame to further representation of the larger agile community as well as U.S.-based projects. Brown's (2015) research used a sample of participants, including IT project managers in the U.S. with experience using various project management methodologies. Brown (2015) recommended further research focus on global companies and a specific agile software development methodology to obtain a better understanding of specific methods and their use in global organizations. Other scholars (e.g., Khalil & Khalil, 2016; Muller & Jugdev, 2012) have recommended research to address the knowledge gaps existing in the identification of the CSFs needed for decision-making throughout the adoption and implementation process for agile software development methodologies.

This study is relevant because global companies are advancing the adoption of agile software development methodologies. More specifically, companies are using Scrum methodology to solve product and software development challenges (Ghani et al., 2015; Kaleshovska, Josimovski, Pulevska-Ivanovska, Postolov, & Janevski, 2015; McLaughlin, 2016; Papatheocharous & Andreou, 2014; VersionOne, 2016a). As adoption and utilization of agile software development methodologies have increased, so have the challenges that global companies and practitioners face, which in turn creates gaps and impediments for successful implementation and scaling (Ghani et al., 2015; McLaughlin, 2016; Papatheocharous & Andreou, 2014; VersionOne, 2016a).

Implications for Business Practice

This study examines the significance of Chow and Cao's 12 CSFs by focusing on large and distributed agile software development projects using Scrum methodology in U.S.-based global companies. In his research on the management information crisis, Daniel (1961) found that just receiving data was insufficient as data needed to be relevant in order to make decisions and measure goals. Rockart (1979) discussed the CSF theory and how to decide what information organizations need in order to make decisions and accomplish their goals. In this vein, research by Bullen and Rockart (1981) defined CSFs as the "things that must go right" (p. 7) for companies to grow and meet their goals.

As discussed by scholars, organizations and practitioners that have pioneered the adoption of Scrum methodology have encountered challenges (Muller & Jugdev, 2012; VersionOne, 2016a). By applying the CSFs concept, theory, and method to research on the implementation of Scrum methodology in U.S-based global companies, this study contributes to knowledge on large and distributed agile software development projects. Specifically, this study makes contributions to the identification of areas that need to be right for the method to be efficient while also mitigating some of the usually incurred challenges. Moreover, by examining the significance of CSFs for the implementation of agile software development methodologies, specifically Scrum, this study helps organizations identify areas of the process that need more focus in order to large and distributed agile software development projects better.

Definition of Terms

The following definitions of terms apply to this study.

Agile software development. Agile software development is a set of software engineering/development methods and practices that evolved from the values and principles outlined in the *Agile Manifesto*. These methods focus on: customer satisfaction, self-organizing and cross-functional teams, and process flexibility and adaptability (Agile Alliance, 2016).

Agile software development project. Agile software development projects are complex projects that use an agile approach to deliver working software as quickly as possible and as requested by the product owner (Optimusinfo.com, 2015).

Business stakeholders. Business stakeholders are groups and individuals who are either internal or external to the company. Business stakeholders include, but are not limited to managers, project team members, subject matter experts, industry groups and associations, and regulatory agencies (PMI, 2013).

Critical success factors (CSFs). CSFs are the few areas needing satisfactory results in order to ensure successful performance of the individual, department, or organization. CSFs are one of the few main areas where success is required for the business to flourish and for the attainment of a managers' goal (Bullen & Rockart, 1981).

Critical success factor (CSF) theory. When rooted in the information systems management field, the CSFs theory seeks to identify and explain the few crucial areas (CSFs) that lead to managerial or organizational success. Practitioners have applied this theory to the direction of an organization's strategic planning, implementation of a plan, execution of a project, and leveraging the performance of a process or organization (Boynton & Zmud, 1986).

Distributed software development team. A distributed team refers to a group whose members are not in the same physical location but rather dispersed across multiple geographical locations and time zones (VersionOne, 2016b). Within the context of Scrum methodology, the software development team is a self-organizing and cross-functional group of three-to-nine individuals that develop and deliver the items in the product backlog (Scrum Alliance, 2016a).

Distributed software development project. Distributed software development projects utilize teams spread across multiple geographical locations and time zones, and often globally (Matalonga et al., 2013).

Iterative process. An iterative process consists of many repeated stages, including a feedback loop after a group of phases is completed (Fernandez & Fernandez, 2008). Scrum methodology proposes an iterative process that allows teams to work on and deliver smaller changes on a set interval of time, allowing for quicker use (Fernandez & Fernandez, 2008).

Large software development project. A large software development project determined by the organization based on a combination of financial impact, number of team members, size of deliverables, complexity, and/or timeframe (Burgan & Burgan, 2014; Hastie & Wojewoda, 2015; PMI, 2013. The project management literature does not provide a single definition that meets the particularities of all organizational and project contexts (Burgan & Burgan, 2014; Hastie & Wojewoda, 2015; PMI, 2013).

Organizational factors. Within the context of agile software development, organizational factors include management commitment, organizational environment, and team environment (Chow & Cao, 2008).

People factors. Within the context of agile software development, people factors include team capability and customer involvement (Chow & Cao, 2008).

Process factors. Within the context of agile software development, process factors include project management process and project definition process (Chow & Cao, 2008).

Product owner. Within the framework of Scrum methodology, the product owner is the individual responsible and accountable for managing the product backlog in a software development project (Scrum Alliance, 2016a).

Product backlog. Within the context of agile software development, the product backlog is a prioritized list (highest need first) of requirements (in story form) given by the product owner for the delivery of the product (Scrum Alliance, 2016a).

Project factors. Within the context of agile software development, project factors include project nature, project type, and project schedule (Chow & Cao, 2008).

Project success. Project success is a combination of on time, on a budget, and with a satisfactory result (Hastie & Wojewoda, 2015). This study includes the "perceived success" concept of achievement, which is when success is determined by the opinion of a key person (e.g., the product owner, project manager, or business stakeholder), and measured by four dimensions – Quality, Scope, Time, and Cost (Chow & Cao, 2008).

SAFe. Scaled Agile Framework is a framework used by companies to integrate, scale, and use the agile software development process at the team, program, and portfolio level (scaledagile.com, n.d.).

Scrum. Scrum, one of the most commonly used agile software development methodologies, entails a management and control process that cuts through complexity and focuses on building software that meets business needs. This method allows management and development teams the ability to understand the requirements and required technologies quickly,

in order to deliver working software incrementally and empirically. Scrum itself is a simple framework for effective team collaboration on complex software projects. (Scrum.org, n.d.).

Scrum master. The Scrum master is the person responsible for ensuring the software development team understands and follows Scrum methods, practices, and rules (Scrum Alliance, 2016a).

Sprint. Within the context of the Scrum methodology, a sprint refers to a time-boxed period, lasting up to 30 days of work during which implementation of an increment of product functionality occurs (Scrum Alliance, 2016a).

Technical factors. Within the context of agile software development, technical factors include agile software techniques and delivery strategy (Chow & Cao, 2008).

U.S.-based global company. A U.S.-based global company, or U.S.-based multinational corporation, refers to a business organization that has operations throughout the world but have their headquarters located within the United States. Tran (2016) defined a multinational company as a business organization with activities found in more than two countries, but with limited foreign direct investment. This structure consists of a country of incorporation and the establishment of divisions in foreign countries.

Assumptions and Limitations

The following is a discussion of assumptions and limitations of this study.

Assumptions

Assumptions are those items or bits of information that are thought to be true but not verified, and which could have an impact on the application of the research results (Laanti & Abrahamsson, 2011). In this study, some assumptions of the study are worth discussing. First,

Scrum has become one of the most adopted agile software development methodologies by organizations because it delivers changes quickly and responds to changes in the industry (VersionOne, 2016a). For these reasons, examining the significance of CSFs for agile software development projects using Scrum methodology is relevant. Scholars have identified some of the ways in which organizations are still encountering challenges in the implementation and scaling of the Scrum methodology. Identified problems include a lack of experience and executive support. Also, sponsorship paired with unrealistic goals and plans and a willingness to celebrate before accomplishing all of these in the adoption (Ghani et al., 2015; Intelliware Software Development, 2016; Khalil & Khalil, 2016; McLaughlin, 2016; Papatheocharous & Andreou, 2014; VersionOne, 2016a).

Second, the identification and application of CSFs (Bullen & Rockart, 1981) contribute to the successful implementation of agile software development methodologies, including Scrum, in large and distributed agile software development projects. CSF theory proposes that a few crucial areas of data can lead to managerial or organizational success, and it applies to the success of agile software development projects (Boynton & Zmud, 1986).

Third, previous research conducted by Chow and Cao (2008) and Brown (2015) has examined CSFs for the implementation of agile methodologies in software development projects; however, most of the data obtained pertain to the XP methodology. Expanding on Chao and Cao and Brown's studies, the purpose of this study was to examine the relationships between 12 independent variables (representing possible CSFs for agile software development projects) and the dependent variable of project success (consisting of four dimensions) in large and distributed software development projects using Scrum methodology in U.S.-based global companies.

Fourth, this study followed an explanatory, quantitative, and survey design in order to measure independent and dependent variables and test hypotheses. The use of quantitative methods, including graphical and quantitative techniques, is appropriate for examining variables and their relationships, testing hypothesis, and answering the study's research questions (Creswell, 2014).

Fifth, the use of a survey instrument served to collect data for the study. This survey instrument has been validated and used successfully for measuring the study constructs in previous research by Chow and Cao (2008) and Brown (2015).

Finally, principles outlined in the *Belmont Report's* (U.S. Department of Health & Human Services, 1979) and guidelines established by the Capella University's Institutional Review Board (IRB) served to ensure the anonymity, privacy, and confidentiality of participants in the study. Following these principles and guidelines, the study's research plan and methods contributed to manage bias, prevent researcher error, and ensure an ethical conduct.

Limitations

Limitations are the potential weaknesses or gaps in a research study (Laanti & Abrahamsson, 2011), and which could impact the generalization and application of its results. Some limitations of the study are worth discussing. First, the study focused only on large and distributed agile software development projects using Scrum methodology – and not all agile software development methodologies. An adaptation of Chow and Cao (2008) and Brown's (2015) survey was needed to address the particularities of Scrum methodology, and it also served to collect the study data.

Second, a self-reporting survey method was used to collect data from participants. Chow and Cao (2008) noted that the utilization of a self-reporting method might introduce bias, as

some participants may either report success on projects that were not successful or overlook challenges encountered during the adoption of agile methodologies in software development projects.

Third, like in other quantitative studies, data collection methods used in this study were dependent on an instrument that is not 100% accurate or reliable (Deutskens, Ruyter, Wetzels, & Oosterveld, 2004). Participants are asked to meet criteria to participate, but it is left to the participant to determine if they met said criteria.

Fourth, participants in the study represent multiple organizations, which in turn provides a representation of the adoption and use of Scrum methodology in large and distributed agile software development projects in U.S.-based global companies. Although representative, this is only a small sample of the population of practitioners using Scrum methodology.

Organization for Remainder of Study

This study is organized into five chapters. Chapter 1 consists of an introduction to the study, including background, business problem, research purpose, research questions, rationale, theoretical framework, and significance, the definition of terms, and assumptions and limitations. Chapter 2 presents a review of the literature concerning agile software development methodologies and CSFs for agile software development projects, with a focus on Scrum methodology. Chapter 3 provides an overview of the research methodology and, more specifically, how quantitative research methods are applied to answer the study's research questions. Chapter 4 summarizes the results of the study, and Chapter 5 discusses implications and recommendations.

CHAPTER 2. LITERATURE REVIEW

Introduction

Expanding on Chao and Cao (2008) and Brown's (2015) studies, the purpose of this study was to examine the relationships between 12 independent variables (representing possible CSFs for agile software development projects) and the dependent variable of project success (consisting of four dimensions) in large and distributed software development projects using Scrum methodology in U.S.-based global companies. The 12 independent variables are management commitment, organizational environment, team environment, team capability, customer involvement, project management process, project definition process, agile software techniques, delivery strategy, project nature, project type and project schedule. The dependent variable of project success consists of four dimensions – Quality, Scope, Time, and Cost.

This chapter provides a comprehensive review of the extant research and literature on project management, agile software development, Scrum methodology, and CSF theory, which serves as the background and the theoretical framework of the study. The literature review includes scholarly seminal and contemporary works published in peer-reviewed journals, conference papers, books, specific book chapters as well as practitioner articles published in journals, magazines, and websites. Online search engines, including Google Scholar, Summon, and databases within Capella University Library (including ABI/INFORM Global, Business Source Complete, and ProQuest Dissertations & Theses Global), provided sources for the literature review. Search strings included terms such as agile, project management, CSFs,

distributed agile, Scrum, adoption of agile, and challenges of agile. The analysis included sources that considered CSF theory, CSFs in project management, CSFs in software development, agile software development, and the Scrum methodology.

TPM no longer addresses the needs of global companies because it is inflexible to the execution of large, emergent, and complex software development projects in distributed environments. For global companies to address the limitation of TPM, they have been adopting agile methodologies, specifically scrum. The Scrum methodology has become the most popular agile methodology within the last decade (Ghani et al., 2015; Kaleshovska et al., 2015; McLaughlin, 2016; Papatheocharous & Andreou, 2014; VersionOne, 2016a). While Scrum methodology have risen in popularity, there are still problems in its implementation in large and distributed software development projects. These challenges are the result of unsupportive leadership and team member inexperience in using the new methodology. Other challenges are time differences, and a lack of communication among co-located teams (Ghani et al., 2015; Intelliware Software Development, 2016; Khalil & Khalil, 2016; McLaughlin, 2016; Papatheocharous & Andreou, 2014; VersionOne, 2016a).

Scholars and practitioners in the field of software development projects and agile methods have applied CSF theory in order to identify areas that need to be accomplished before achieving success (Brown, 2015; Chow & Cao, 2008; Khalil & Khalil, 2016; Muller & Jugdev, 2012). For instance, Chow and Cao (2008) conducted a quantitative, non-experimental, survey study to explore CSFs for agile software development projects. They consolidated a list of 12 possible CSFs and found that only a small number (a correct delivery strategy, a proper practice of agile software engineering techniques, and a high-caliber team) were significant predictors of project success (consisting of four dimensions – quality, scope, time and cost). They

recommended further research to re-test their model in order to find "whether any new factors may emerge or current key success factors become no longer critical" (p. 968) as well as addressing some limitations about their study's sample size, representation of U.S.-based projects, and bias towards XP projects.

Brown (2015) replicated Chow and Cao's (2008) study by addressing the limitation of minimal U.S.-based project representation. Brown found that only "six factors were significant: project type, schedule, project nature, management commitment, definition process, and delivery strategy" (p. 103). Only one of these factors, delivery strategy, was included in both results from Brown (2015) and Chow and Cao, which may be explained by "the immature state of the agile development methods" as explained by Chow and Cao. Brown recommended additional research concentrating on specific agile software development methodologies.

This study contributes to the existing body of knowledge in the field of project management by identifying how the CSFs for agile software development projects continue to be a relevant topic of inquiry. Additional research is needed given the low rate of success of large and distributed agile software development projects and changing trends within the software development industry over the past decade (Ghani et al., 2015; Kaleshovska et al., 2015; McLaughlin, 2016; Papatheocharous & Andreou, 2014; VersionOne, 2016a). One of these trends is the growing popularity of the Scrum methodology (Ghani et al., 2015; Kaleshovska et al., 2015; McLaughlin, 2016; Papatheocharous & Andreou, 2014; VersionOne, 2016a). The organization of this chapter is in four main sections, including agile software development methods, Scrum methodology, CSF theory, and a conclusion. Each of the first three sections presents a discussion and blending of relevant seminal and contemporary research studies, which are further synthesized and integrated into the concluding section.

Agile Software Development Methods

Before the early 2000s, TPM was the preferred approach for companies of all sizes to introduce organizational change. Since then, agile methods have emerged as an alternative approach to dealing with more emergent and complex endeavors, including software development projects. Agile software development refers to a set of methods and practices aligning with the principles put forward in the *Agile Manifesto*, including those used by self-organizing teams that collaborate to drive evolving solutions (Agile Alliance, 2016). This section addresses the transition from TPM to agile software development methods, provides an overview of agile software development, and compares TPM, and agile software development approaches.

From TPM to Agile Software Development Methods

The theory of project management has been in existence and application since before the fourteenth century. Kozak-Holland and Procter (2013) found early references to the use of project management techniques in the construction of the Dome of the Florence Cathedral in Italy during the thirteenth century. Grant and Kelly (2009) found that, many centuries later, coordination efforts were used to build structures by using a master builder or overseer to direct the work of others. Then, in the twentieth century, project management began to expand with Frederick Taylor using a break-down approach of productivity to eliminate extra steps in a process (Gantt.com, 2016). And Henry Gantt creating the technique known now as the Gantt chart, which addresses the sequencing of tasks (Gantt.com, 2016). Project management continued to grow in popularity when in 1958 the U.S. military introduced PERT, a method for determining the timing of tasks and overall projects (Interventions, 2015). Moreover, the formalized introduction of the practice of project management came with the advancement of the

digital revolution and management practices of the second part of the twentieth century (Interventions, 2015).

As project management practice grew in popularity, various professional associations and member organizations began to establish certifications. Alexander (2015) and Florentine (2015) have noted that with the growth of project management as a profession, various professional associations and member organizations have been established around the world to advance the body of knowledge, provide training and certification, and support the work of practitioners. Among these are the Project Management Institute (PMI), the International Association of Project Managers (IAPM), the International Association of Project and Program Management (IAPPM), and the American Academy of Project Management (AAPM) (Alexander, 2015). Similarly, as Alexander (2015) and Khorrami Rad (2014) found, there are several professional certifications recognized in the industry and used for validating the competency of project management professionals. These certifications include: (a) the Project Management Professional (PMP)® and the Certified Associate Project Manager (CAPM)®, issued by the PMI; (b) the Certified Project Manager (CPM), issued by the IAPPM; (c) the Master Project Manager (MPM)®, issued by the AAPM; (d) the CompTIA Project+, issued by the CompTIA; and (e) the Projects IN Controlled Environments (PRINCE2)®, issued by AXELOS.

Among various professional associations/member organizations and certifications for project managers, the PMI and its PMP® respectively, have become the most popular around the world, with the most members and certified professionals found in the U.S. (Rongala, 2015). In 1969, the PMI was one of the first professional association and member organization to focus on research and practice in the field of project management; it has now grown to include chapters in over 80 countries (PMI, 2017). *A Guide to the Project Management Body of Knowledge*, known

as the *PMBOK® Guide*, now in its fifth edition (PMI, 2013), has become the global standard for project management, addressing fundamental concepts, practices, and tools for achieving organizational results and excellence in projects.

As defined in the *PMBOK® Guide* (PMI, 2013), a project is "a temporary endeavor undertaken to create a unique product, service, or result, that has a definite beginning and end" (p. 1). Also, project management refers to "the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements" (p. 5). As outlined in the *PMBOK® Guide*, there is a set process recommended for successful completion and management of a project sequentially, from beginning to end, commonly referred as TPM. As part of the TPM process, managing a project includes, but is not limited to, working with stakeholders to identify and manage requirements, needs, challenges, and expectations. This process involves meeting the scope/quality, time/schedule, and budget/cost/resource expectations of stakeholders, more commonly referred as "the triple constraints of projects" (Bannerman, 2008; PMI, 2013).

Scholars and practitioners have described TPM as a very structured, clear, and hierarchical approach requiring a high level of effort for documentation and analysis/design of a product in preparation for execution (and actual software development, in the case of software development projects) (Fernandez & Fernandez, 2008; PMI, 2013). The *PMBOK® Guide* (PMI, 2013) described the TPM's as a phased approach that can be (a) used for all projects; and (b) applied in other variations, such as using it multiple times in sequence depending on the needs and size of a project. The TPM method proposes managing a project against an agreed-upon budget, schedule, and scope of work (Fernandez & Fernandez, 2008). TPM also assumes that processes and documentation must occur in a specific order, with each process and its related

documentation completed before moving on to the next (Hoda & Murugesan, 2016; PMI, 2013). The project lifecycle proposed by the TPM approach includes the "processes of initiating, planning, executing, monitoring and controlling, and closing," which together form a project phase (PMI, 2013, p. 42). The planning and executing processes combine to create the monitoring and controlling processes and are repeating until the project moves into the closing processes.

As previously noted, TPM focuses not only on the processes of a project but also on the tasks and documentation required. As illustrated by the PMI (2013), before a project can even begin, TPM expands a high level of effort and time on tasks and the documentation of various processes in the lifecycle of a project (PMI, 2013). Each process has associated tasks and required documentation that is used to: communicate the overall health of the project; highlight accomplishments, risks, or issues; and build and sustain buy-in from the stakeholders throughout the project's lifecycle. The PMI (2013) also illustrated some of the documents required as part of the process, including critical information such as the scope, requirements, budget, stakeholders, work breakdown structure, designs, and testing criteria. Throughout the project lifecycle, checkpoints and document approvals occur at specified times, which in turn serves to report the status of accomplishments to stakeholders and obtain approval to continue the project. These checkpoints or milestones are intended to ensure clarification of scope, identify any risks or challenges, and check on the validity and approval for a continuance of the project.

Even though the TPM approach has been useful over the last four decades, its usefulness is called into question when compared to the level of complexity and uncertainty of modern software development projects. Many scholars have argued that TPM's one-size-fits-all approach no longer lends value to organizations (Cooke-Davies, Crawford, & Lechler, 2009;

Fernandez & Fernandez, 2008; Sargut & McGrath, 2011; Saynisch, 2010a). The TPM approach requires a significant investment of time before delivery of any actual product; a process that does not serve organizations seeking a more expedient delivery of results for their global and increasingly more complex projects. Furthermore, when using TPM, organizations commonly encounter barriers such as differences in communication, culture, language, work ethics, and time zones (Anantatmula & Thomas, 2010; Hanisch & Corbitt, 2007)

Projects managed using the TPM approach face significant challenges and have a high rate of failure. For instance, results from the CHOAS Report (as cited in Hastie & Wojewoda, 2015) found that of projects that are managed using the TPM approach, 29% fail and 60% face significant challenges. On the other hand, of those managed using agile methods, only 9% fail and 52% were confronted with significant challenges (as cited in Hastie & Wojewoda, 2015). Though the TPM approach is challenging and has a higher rate of failure, it remains relevant and valuable as an option for specific kinds of projects (Hastie & Wojewoda, 2015). However, in other projects where flexibility and expedience are needed, the TPM approach is not a good fit.

As information and communication technologies have become more widely used, organizations and practitioners have called for new approaches to project management that deliver products faster and require less documentation (Campanelli & Parreiras, 2015; VersionOne, 2016a). As a result, the creation of agile methods, processes, and methodologies caused the development of the *Agile Manifesto*.

An Overview of Agile Software Development

Agile software development refers to "an umbrella term for a set of methods and practices based on the values and principles expressed in the *Agile Manifesto*. Solutions evolve through collaboration between self-organizing, cross-functional teams utilizing the appropriate

practices for their context" (Agile Alliance, 2016, What is Agile Software Development? section, para. 1). This approach to software development has emerged as an alternative to the TPM approach, with the aim of delivering projects in a more iterative, flexible, and agile fashion, and without the heavy documentation requirements entailed by a traditional project lifecycle (Dyba & Dingsoyr, 2008).

The *Agile Manifesto* emerged from a group of independent practitioners in software engineering/development that shared an interest in advancing agile software development methodologies. The people behind the *Agile Manifesto* included "representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others sympathetic to the need for an alternative to documentation driven, heavyweight software development processes" (para. 1). Published in 2001, the *Agile Manifesto* laid out common principles for agile software development, and regardless of the mechanics of any specific methodology, it advocates:

1. Our highest priority is to satisfy the customer through an early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals.
6. Give them the environment and support they need, and trust them to get the job done.
7. The most efficient and effective method of conveying information to and within a development team is a face-to-face conversation.
8. Working software is the primary measure of progress.
9. Agile processes promote sustainable development.
10. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
11. Continuous attention to technical excellence and good design enhances agility.
12. Simplicity – the art of maximizing the amount of work not done – is essential.
13. The best architectures, requirements, and designs emerge from self-organizing teams.

14. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. (p. 1)

These principles underline the aim of agile software development methodologies and their practitioners for (a) delivering working products quickly; (b) being flexible; and (c) stressing the role and accountability of self-organizing teams and their members.

The *Agile Manifesto* led to the birth of the Agile Alliance, the primary organization for the agile software development community (Agile Alliance, 2016). The Agile Alliance "supports people who explore and apply agile values, principles, and practices to make building software solutions more efficient, humane, and sustainable. We share our passion for delivering software better every day" (Agile Alliance, Mission Statement section, para. 1).

Agile Software Development Methodologies

A myriad of methodologies has emerged as a part of the agile movement. These methods share the 12 principles outlined in the *Agile Manifesto* that includes, but are not limited to, Scrum, XP, Lean, Kanban, FDD, DSDM, ASD, Crystal, and RUP (Brown, 2015; Chow & Cao, 2008; Campanelli & Parreiras, 2015; McLaughlin, 2016). First, the Scrum methodology "is a simple yet incredibly powerful set of principles and practices that help teams deliver products in short cycles, enabling fast feedback, continual improvement, and rapid adaptation to change" (Scrum Alliance, 2016b, para. 1). The Scrum methodology is a repetitive process starting with the product owner creating a product backlog (Scrum Alliance, 2016c). The team chooses to work on the highest priority items in the backlog in a time-box period called a sprint. The plan is to deliver a working product to the product owner at the end of a sprint (or multiple sprints), after which the process repeats itself (Scrum Alliance, 2016c).

Second, the XP methodology is used for a team to deliver high-quality software continuously, while also maintaining a high level of customer involvement, quick feedback loops, continuous testing, and planning (Lindstrom & Jeffries, 2004; McLaughlin, 2016). The goal in XP software development is to deliver quality software at frequent intervals, usually every one-to-three week(s) (Lindstrom & Jeffries, 2004; McLaughlin, 2016).

Third, the Lean methodology is an "iterative, agile methodology that focuses on the team delivering value to the customer, and on the efficiency of the 'Value Stream' or the mechanisms that deliver that value" (McLaughlin, 2016, para. 4). As described by McLaughlin (2016), the focus of the Lean methodology is to eliminate waste by delivering only valuable, prioritized features in small batches while also empowering the team to gather rapid and reliable feedback from programmers and customers. Lean aims to both maximize team productivity with concurrent work while also minimizing workflow dependencies. In this method, the team pulls work based on the customer's request, which allows them to concentrate on the product's most valuable features.

Fourth, in the Kanban methodology, teams partner with one another in order to manage the development of working products and focus on continuous delivery while also avoiding overburdening themselves (McLaughlin, 2016). This methodology has three simple principles: "(a) visualize what you do today (workflow); (b) limit the amount of work in progress (WIP); and (c) enhance flow" (para. 8). The Kanban methodology allows for quick and continual delivery because teams pull work from a prioritized set of requirements.

Fifth, the FDD methodology proposes "a client-centric, architecture-centric, and pragmatic software process" (Ambler, 2014b, para. 1). Ambler further explained that a "feature," or an "a small, client-valued function expressed in the form

<action><result><object>" is a central component of the method (para. 2). FDD uses an iterative process consisting of five main activities. The activities are: (a) start with the development of an overall high-level model or design; (b) create a features list grouped by sets; (c) plan owners by feature; (4) develop; and (5) build the features (Ambler, 2014b).

Sixth, the DSDM methodology uses nine key principles. The principles consist of: (a) business needs/value, (b) active user involvement, (3) empowered teams, (4) frequent delivery; (5) integrated testing, and (6) stakeholder collaboration (Gupta, n.d.; McLaughlin, 2016). DSDM has planning and delivery of requirements in short and fixed time-boxes, with requirements prioritized and ranked by what must be, should be, could have, and will not have at this time (McLaughlin, 2016). This framework can be independent or used in conjunction with other iterative methodologies.

Seventh, the ASD methodology promotes an incremental and iterative development process for large and complex systems by using constant prototyping and feedback (Abrahamson, Warsta, Sippon, & Ronkainen, 2003). Highsmith (2002), one of the creators of ASD, noted that the ASD methodology proposes "a life cycle dedicated to continuous learning and oriented to change, re-evaluation, peering into an uncertain future, and intense collaboration among developers, management, and customers" (p. 7).

Eight, Crystal refers to a family of agile methodologies such as Crystal Clear, Crystal Yellow, and Crystal Orange, which focus on factors such as team size, system criticality, and project priorities (McLaughlin, 2016). As described by McLaughlin (2016), the Crystal methodology is a lightweight, adaptable method for software development, which can be modified based on the needs of different projects. As teams use the methodology, they may make modifications and communicate as needed for simplicity and flexibility.

Finally, the Rational Unified Process (RUP) methodology "provides a disciplined approach to assigning tasks and responsibilities within a development organization to deliver high-quality software that meets the needs of the end-user" (IBM, 1998, para. 1). IBM (1998) further noted that RUP consists of four different phases, each of which concludes with a milestone: inception, elaboration, construction, and transition.

A survey conducted by VersionOne (2016a) found that of these varied agile software development methodologies, XP and Scrum are the most popular. The "10th Annual State of Agile Report" (VersionOne, 2016a) showed that of 4,000 participants, 58% used Scrum methodology, and an additional 10% used a hybrid of Scrum and XP methodologies, with many of the projects conducted in a distributed team environment. Other surveys have shown that the Scrum methodology is growing in popularity (Kaleshovska et al., 2015; McLaughlin, 2016; Papatheocharous & Andreou, 2014; VersionOne, 2016a).

Comparing the TPM and Agile Software Development Approaches

Dyba and Dingsoyr (2009) compared the perspectives of TPM and agile approaches when applied to the software development process. The TPM perspective is deliberate and formal, so long as the system environment is stable and predictable. In contrast, the agile perspective is emergent, iterative, and exploratory, mirroring a system environment that is turbulent and difficult to predict. Moreover, in creating a system or developing software, the TPM approach may be best utilized to develop a comprehensive build plan. However, if a system or software requires modification, the agile approach would be better, as it can deliver the product quickly and react to emergent and continuous change.

An important point of difference between the TPM and agile approaches is the team and management structure. The *PMBOK® Guide* (PMI, 2013), described TPM as a formal

organization structure governed by hierarchical lines of reporting that are established by the administration at the beginning of the project (PMI, 2013). In contrast, as noted by McLaughlin (2016), agile teams are smaller and self-governing. In this approach, the product owner, who is also part of the team, is responsible for driving product requirements.

Studies have documented the processes organizations, and practitioners use to transition from TPM to agile methods. Research has found that there are several different strategies for transitioning from TPM to agile methods. Also, there are consistent and more positive results when companies transition from the hierarchical TPM approach to the more flexible and collaborative agile approach (Dikert, Paasivaara, & Lassenius; 2016; Dyba & Dingsoyr, 2008; Fernandez & Fernandez, 2008; Laanti & Abrahamsson, 2011).

Dyba and Dingsoyr's (2008) literature review of agile software development consisted of 36 studies organized into four categories: "introduction and adoption, human and social factors, perceptions of agile methods, and comparative studies" (p. 7). They conclude that the studies evaluated had limitations, "such as the unsustainability of the on-site customer's role for long periods and the difficulty of introducing agile methods into large and complex projects" (p. 8). Even with these limitations, Dyba and Dingsoyr suggested focusing on human and social factors along with staffing agile teams with people that believe in their abilities and good interpersonal skills.

Fernandez and Fernandez's (2008) review of the literature highlighted some of the different strategies used by the TPM and agile approaches. The scholars compared several strategies for transitioning from TPM to agile methods, including (a) linear, (b) incremental, (c) iterative, (d) adaptive, and (e) extreme. Fernandez and Fernandez (2008) found differences vary

by phases per cycle and scope readjustments based on feedback from experience with each of the project iterations.

Laanti and Abrahamsson's (2011) survey of large-scale agile transformation within a global company illustrated the effects of agile transformation. At the time of the study, 50% of the participants had positive responses to the use of agile methods. Survey participants reported that using agile methods led to a greater sense of personal satisfaction and effectiveness. Participants in this survey also believed that agile methods resulted in a better product because of identification of product defects early in the process.

Finally, Dikert et al. (2016) documented the complexities of using differing methodologies within an agile software development project. The scholars reported that challenges are likely to occur when a project involves multiple groups, each with their understanding of output, effort, and agile methodology. Dikert et al. further noted that creation of additional challenges occurs when project managers attempt to ensure product quality by requiring that multiple groups produce additional documentation on their methodologies.

Summary

Even though TPM has a foundation back to the fourteenth century, it frequently does not meet the needs of modern day organizations with software development projects. This change is partly because of global expansion, increased complexity, and uncertainty in the projects (Ghani et al., 2015; Kaleshovska et al., 2015; McLaughlin, 2016; Papatheocharous & Andreou, 2014; VersionOne, 2016a). In 2001, agile software development practitioners developed the *Agile Manifesto* in an attempt at documenting and advancing universal principles and practices shared by emerging agile methodologies. This change helped meet the needs of businesses and deliver products more quickly. The vision of the *Agile Manifesto* includes prioritizing people, teams,

and customers, and minimizing required project documentation. The principles set by the *Agile Manifesto* stressed that: (a) the client is the highest priority, (b) change is welcomed, and (c) people must work together to deliver products quickly—all while keeping the process simple.

Extant research (e.g., Chow & Cao, 2008; Brown, 2015) has found that while XP has previously been one of the more popular agile methodologies, this has changed. The Scrum methodology has now grown to become the most popular choice for large global organizations (Ghani et al., 2015; Papatheocharous & Andreou, 2014; Scrum Alliance, 2016c; VersionOne, 2016a). The Scrum methodology offers structure that other agile methods lack, which organizations need as they transition to agile (VersionOne, 2016a). Given its popularity and expected growth in use, this study focuses on the CSFs for large and distributed agile software development projects using Scrum methodology.

Scrum Methodology

Scrum, one of the agile methodologies supported by the *Agile Manifesto*, emphasizes the critical role of individual team members and customer collaboration throughout the software development process. Scrum has quickly become one of the most popular agile methodologies, especially in global organizations (Laanti & Abrahamsson, 2011; VersionOne, 2016b). This section addresses the Scrum methodology, including its history and the ways in which it is challenging to implement.

Overview of the Scrum Methodology

Scrum methodology is designed to be lightweight framework allowing for quick delivery, risk reduction, and flexibility (VersionOne, 2016b). Schwaber and Sutherland (2016) developed the methodology in the early 1990s as a "framework for developing and sustaining complex

products" (para. 1). The method consists of various teams, events, artifacts, and rules that serve a purpose to the overall software development process.

The foundation of Scrum methodology lies in empirical process control theory, or empiricism (Schwaber & Sutherland, 2016). First, transparency proposes the need for all observers and participants to share the same product view and understanding of what "done" means. Second, inspection requires establishing progress points throughout the software development process to ensure detection of undesirable variances in work. Lastly, adaptation refers to being flexible and adjusting as soon as possible to account for deviations outside the acceptable limits minimizing impacts in the overall software development process.

Schwaber and Sutherland (2016) argued that Scrum methodology requires teams and others involved in the development process agree to honor the values of commitment, courage, focus, openness, and respect. Additionally, the team needs to develop a good understanding of what the methodology entails, with its various concepts, events, ceremonies, and artifacts. Accordingly, the following discussion explains the different aspects of the Scrum methodology, including the team, software development cycle, events and ceremonies, and artifacts.

Scrum team. A Scrum team is a group of individuals that must be cross-functional and self-organizing. Members of the team, as a group, decide how to accomplish the tasks best, and must have the needed experience and knowledge to complete the required work without depending on others. This independence allows the team to be flexible, creative, and productive (Deemer, Benefield, Larman, & Vodde, 2012; Schwaber & Sutherland, 2016).

The Scrum team is a group of individuals represented by three primary roles: (a) product owner, (b) Scrum master, and (c) team members (Deemer et al., 2012; McLaughlin, 2016; Schwaber & Sutherland, 2016; Sharp & Ryan, 2011). First, one person holds the role of product

owner, which is responsible for identifying and prioritizing the features or scope of the product. The product owner needs to be respected and supported by team members to have the capacity to make decisions about priorities (Scrum Alliance, 2016d; McLaughlin, 2016). Schwaber and Sutherland (2016) argued that the product owner is responsible for ensuring that the backlog is clear, visible, and transparent so the team can easily choose items. The product owner is also the person who reviews and accepts the delivered product, and reprioritizes backlog items as needed (Deemer et al., 2012).

Second, the Scrum master guides the team and ensures that they understand and adhere to the Scrum theory, practices, and rules (Schwaber & Sutherland, 2016; Scrum Alliance, 2016d). The Scrum master serves as a coach or servant-leader, so that team members learn and apply Scrum theory, practices, and norms for best results. However, it is important to note that the Scrum master is not a manager, but rather a coach/teacher for two groups – the team and the overall organization (Deemer et al., 2012; Schwaber & Sutherland, 2016; Scrum Alliance, 2016d). The Scrum master works with the team to facilitate whatever is needed. The facilitation may include (a) removing impediments, (b) helping the team to adopt Scrum, (c) educating and coaching, (d) working with the product owner to manage the backlog, and (e) facilitating Scrum events and ceremonies. Organizationally the goals of the Scrum master are to (a) lead and coach through Scrum adoption, (b) help those within the organization understand and practice Scrum, (c) improve the team practices and productivity, and (d) work with other Scrum masters to improve Scrum utilization and effectiveness.

Third, the Scrum team is a self-managing group of five to 12 people who possess the skills and experience needed to deliver a quality and shippable product during each sprint (Deemer et al., 2012; Schwaber & Sutherland, 2016; Scrum Alliance, 2016d). A team needs to

be small enough to be flexible, yet large enough to complete enough work within a sprint (Deemer et al., 2012; McLaughlin, 2016; Schwaber & Sutherland, 2016; Scrum Alliance, 2016d). Overall, the Scrum team is accountable for the delivery of a quality product (Scrum Alliance, 2016d), and although each team member may not have a specific job title, they perform one or more of the following roles: business analyst, developer, and architect (Deemer et al., 2012). The Scrum team may be self-managing and experienced; however, team members escalate impediments as they arise so that management can help mitigate them (Deemer et al., 2012; McLaughlin, 2016).

The *Agile Manifesto* and other sources (e.g., Deemer et al., 2012; McLaughlin, 2016; Schwaber & Sutherland, 2016; Sharp & Ryan, 2011) have suggested that co-located teams produce the best results. However, virtual and distributed teams can also be successful. As organizations expand globally, agile teams sometimes co-locate a factor that can create more challenges to overcome to achieve success. For example, a distributed group that is across many locations must still be able to interact. This situation is a factor that can be mitigated through the use of online tools and conference calls (Deemer et al., 2012; McLaughlin, 2016; Schwaber & Sutherland, 2016; Scrum Alliance, 2016d; Sharp & Ryan, 2011). Distributed Scrum teams, whether co-located or working remotely, must embrace the values set out in the *Agile Manifesto* and work hard to ensure open communication (Sharp & Ryan, 2011). Finally, in addition to the product owner, Scrum master, and team members, other roles play a significant part in the software development process and can have a significant impact on the success of the product. These functions include managers, customers, and end-users, who support the team, remove impediments, and provide expertise and experience (Deemer et al., 2012).

Scrum software development cycle. The Scrum software development cycle is an iterative and incremental process that meets the emerging needs of end users by focusing on adaptability and flexibility (Schwaber & Sutherland, 2016; Scrum Alliance, 2016d). The Scrum software development cycle occurs in a time-box called a sprint, which can last from one-to-four weeks, with two weeks being the usual norm. However, depending on the size of the product and its deadline, the period of development can cross multiple sprints (Deemer et al., 2012; McLaughlin, 2016; Schwaber & Sutherland, 2016; Scrum Alliance, 2016c; Sharp & Ryan, 2011). An established cadence is the most important aspect of the Scrum software development process. This rhythm helps allow the team to operate as a group, not just as individuals. The process continually holds the team accountable for each other while also working to progress toward the delivery of the product (Agile Alliance, 2016; Scrum Alliance, 2016c; VersionOne, 2016b).

Figure 2 illustrates steps in the Scrum software development cycle as reflected in many different aspects of research (Cao, 2006; Scrum Alliance, 2016c; VersionOne, 2016b). First, the team establishes a sprint or set time-boxed period, usually two-to-four weeks in length. Second, the team works with the product owner to determine what items to include in the backlog in the upcoming sprint. Third, the team begins the sprint and development of the product with a daily meeting. This meeting is called a stand-up, which occurs at the beginning of each day, usually lasting 15 minutes or less. The standup meeting is used to discuss progress, impediments or questions that team members may have for meeting the deadline and details expectations of the sprint. Fourth, at the end of the sprint, the product owner evaluates and accepts the delivered product and reviews the team (Agile Alliance, 2016; VersionOne, 2016b). Fifth, at the end of the sprint, the team also holds a retrospective conversation to discuss what did and did not go

well in the sprint, as well as what are the needed improvements for the next sprint (Agile Alliance, 2016; VersionOne, 2016b).

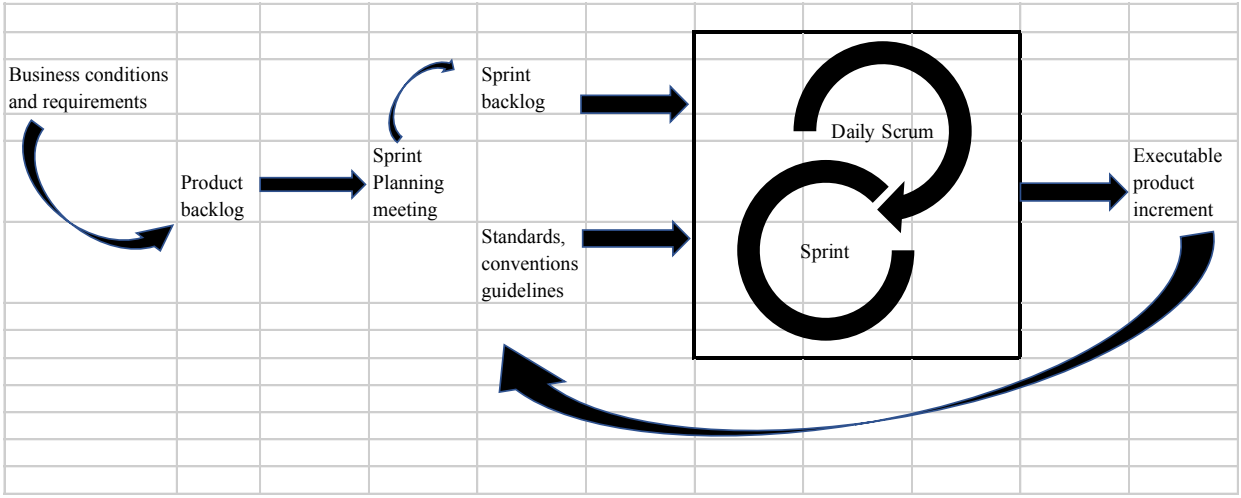

Figure 2. The Scrum software development cycle reflects the scrum process from identification of requirements and backlog creation through to delivery of a product.

Scrum events and ceremonies. Scrum events and ceremonies provide regularity and consist of quick meetings at fixed intervals. These meetings serve to minimize the need for additional random meetings outside of the established routines. Scrum events and ceremonies also serve to ensure transparency and facilitate the product owner's constant inspection and adaptation. Failure to include any of these events and ceremonies in the software development process can reduce transparency and create risks (Schwaber & Sutherland, 2016; Scrum Alliance, 2016c). Scrum events and ceremonies are time-boxed to establish a maximum duration while ensuring quick, concise communication, and eliminating wasted time (Schwaber & Sutherland, 2016; Scrum Alliance, 2016c).

These events and ceremonies include three groupings that occur before, during, and after the sprint. Before the sprint starts, the following occur: sprint time-box determined, the definition of done, and the sprint planning. Next, during the sprint, the daily Scrum, the product

backlog refinement, and tracking progress during the sprint occur. Lastly, the sprint review and the sprint retrospective allow for the wrapping-up of one sprint and moving on to the next (Deemer et al., 2012; Schwaber & Sutherland, 2016).

First, as discussed, the sprint, or a time-box ranging from two-to-four weeks, is the "heart" of the Scrum process because it is the way that a "done," usable, and possibly releasable product happens (Schwaber & Sutherland, 2016; Scrum Alliance, 2016c). The sprint is a consistent and repeatable routine, and once it has started, changes to the project are not allowed without agreement and the acceptance of risk from the product owner (Schwaber & Sutherland, 2016; Scrum Alliance, 2016c). Only a product owner can cancel a sprint, at which time, the goal is no longer relevant. However, sprints are rarely canceled because they are short and flexible, which allows for changes in priorities (Deemer et al., 2012; Schwaber & Sutherland, 2016). Based on the requirements and backlog, the product owner determines the number of sprints required to complete the project and deliver the final product (Deemer et al., 2012).

Second, the establishment of the next most important event before work begins is an agreement from all team members of the definition of "done." As Deemer et al. (2012) clarified, the definition of done should be as close to the potentially shippable product as is possible; otherwise, there may be delays in delivery of the final product. The definition of "done" can be adjusted, but it sets the standard for the quality and completion of the product (Deemer et al., 2012).

Third, once establishment and agreement of the length of the sprint and the definition of done, the product owner and the team, jointly, prepare for a new sprint with a planning event. This meeting takes place at the beginning of the sprint, usually, the first day, as the team reviews the high-priority items in the product backlog (Deemer et al., 2012). Sprint planning is time-

boxed to a maximum of eight hours for a one-month sprint; however, reduction of planning time applies if there is a shorter sprint period. The goal of the sprint planning event is to answer two questions: (a) what can the team deliver in the time resulting from the upcoming sprint? And (b) how will the required work for delivery of a product increment be completed? (Deemer et al., 2012; Schwaber & Sutherland, 2016). Deemer et al. (2012) and Schwaber and Sutherland (2016) have agreed that the team needs to determine its work time capacity to answer the two sprint planning questions noted above. Then, based on this capability limit, the team reviews the prioritization of the backlog and adds items to the scope of the sprint. After setting the capacity and scope of the sprint, the team examines the requirements in detail to determine which need to be completed for delivery at the end of the sprint (Deemer et al., 2012). At the end of the sprint planning meeting, the product owner and team develop an agreement and commitment to the anticipated scope.

Fourth, the daily Scrum is a 15-minute time-boxed event that allows team members to coordinate efforts amongst themselves to create a plan for the next 24 hours. There are many ways to conduct this event, but the most commonly suggested is for everyone to remain standing and answer three core questions:

1. What did I do yesterday to help the team meet the sprint goal?
2. What will I do today to help the team meet the sprint goal?
3. Do I see any impediment that prevents the team or me from meeting the sprint goal?

The daily Scrum meeting serves as a checkpoint with the team, not a status report (Deemer et al., 2012; Schwaber & Sutherland, 2016; Scrum Alliance, 2016c). In agile software development, if one person fails, the whole team fails. The goal of this event is to improve knowledge and

decision-making skills while also raising identifying challenges that have been encountered and need to be mitigated (Deemer et al., 2012).

Fifth, product backlog refinement is a small, yet important, routine meant to ensure the backlog requirements are ready for the team to start working (Schwaber & Sutherland, 2016). The goal of product backlog refinement is to ensure that requirements are in workable size items with sufficient detail to reprioritize the list based on importance and to prepare stories for future sprints (Deemer et al., 2012). This event only takes about 10% of the team's capacity and requires entire team participation. Product backlog refinement is an ongoing process, rather than an event occurring on a specific day/time.

Sixth, it is crucial during a sprint that Scrum teams track their progress and identify the delivered and pending items (Deemer et al., 2012). Scrum teams use many tools, manual and automated, to help track their progress. Some of these tools are task boards, burndown (completion) charts, and marking stories and requirements as completed to reflect what is still outstanding of the agreed commitment (Deemer et al., 2012; Schwaber & Sutherland, 2016; Scrum Alliance, 2016c).

Seventh, at the end of a sprint, usually the last day, the team conducts an event called sprint review (or demonstration), for the product owner and other interested stakeholders (Deemer et al., 2012; Schwaber & Sutherland, 2016). The sprint review allows the development team to collaborate with the concerned parties and the product owner for an inspection of the product. During this event, the determination of if any changes need to be made before meeting the needs of end users (Deemer et al., 2012; Schwaber & Sutherland, 2016; Scrum Alliance, 2016c). This event is time-boxed to one hour per week of a sprint and requires minimal preparation time.

Finally, the last event of the sprint is the sprint retrospective, which follows the review. This event is an opportunity for team members to plan and reflect before the next sprint (Deemer et al., 2012; Schwaber & Sutherland, 2016; Scrum Alliance, 2016c). Often, Scrum teams skip this event to save time; however, this event is critical for the holistic development of the team. The sprint retrospective identifies what went well, what should stay the same, and what needs to be changed or improved; it is a step toward team improvement, and a time for open communication regarding people, process, tools, and relationships. The time-box for this event is three hours for a one-month sprint and can be shortened or lengthened as needed (Deemer et al., 2012; Schwaber & Sutherland, 2016).

Scrum artifacts. Schwaber & Sutherland (2016) explained that Scrum artifacts are created by the product owner and Scrum master to represent work, and are used by the team to maximize transparency and ensure consistent understanding of requirements. There are two main artifacts in the Scum process: (a) the product backlog, and (b) the sprint backlog (Deemer et al., 2012; Schwaber & Sutherland, 2016; Scrum Alliance, 2016c). First, the product backlog is an ordered list of all requirements needed for the product. It serves as the single source for requirements and changes, is never complete, and can be reprioritized at any time (Deemer et al., 2012; Schwaber & Sutherland, 2016). The product owner owns the product backlog and is responsible for its content, availability, and ordering. The product backlog continues to grow as long the product or system exists, and it typically consists of features, functions, requirements, enhancements, and fixes that change in future sprints. Even when multiple teams are working on the same product, there is still only one product backlog. The product owner and team can monitor progress toward a goal by using various reports, like burn-up, burn-down charts, and/or cumulative flows (Deemer et al., 2012; Schwaber & Sutherland, 2016).

Second, the sprint backlog is a selection of the highest priority items, and it serves as a team map for the next sprint event. The sprint backlog is what the team needs to deliver and is the committed product (Schwaber & Sutherland, 2016).

Challenges for the Adoption of Scrum in Agile Software Development Projects

The "10th Annual State of Agile Report" conducted by VersionOne (2016a) surveyed close to 4,000 respondents, of which a quarter worked for large organizations. In this survey, over 82% worked with distributed teams, and 68% used Scrum or a Scrum hybrid approach. This report supported the assumption that Scrum is one of the most popular agile methodologies used in software development projects.

Scrum has the structure of time-boxed events and requires some standard artifacts; however, even with this structure, teams and organizations encounter challenges during implementation (Deemer et al., 2012; Schwaber & Sutherland, 2016; Scrum Alliance, 2016c). These types of challenges (e.g., management involvement and co-located teams), have increased and change with the growth in usage of the Scrum methodology by global companies for large and distributed agile software development projects. The VersionOne (2016a) survey indicated, most organizational cultures are hierarchical, and as such, are at odds with the flexible and fluid values of agile. As more organizations adopt and implement Scrum methodology, the types and kinds of challenges will increase; however, some challenges will certainly remain: resistance to change, inexperienced team members, and a lack of management support.

Most research asserts that Scrum methodology works for companies of all sizes. However, there is a dire need for better and more disaggregated research that can better account for the unique challenges faced by different types of organizations (Dikert et al., 2016; Hoda & Murugesan, 2016; Papatheocharous & Andreou, 2014). For example, in their quantitative study

on agile methods, Papatheocharous & Andreou (2014) found that among 377 participants represented, 76% were using Scrum, and of those, 50% with three to10 projects, and 26% of more than 20 projects. These findings offer support to the argument that the benefits of agile consist of (a) accelerated time to market, and (b) management and prioritization of changing requirements. Khalil and Khalil's (2016) found that while companies are increasingly adopting agile methods, they are not modifying business objectives or the organizational structure to support the values behind the approach. It needs to be noted, however, that Khalil and Khalil's framework is based on the research of Weill and Ross (an IT governance model), and has not been used or tested within the context of the field of project management.

Ambler (2014a) and Chikhale and Mansouri (2015) have argued that changes to an approach or process always bring the possibility of misunderstanding concepts or theories. For example, there is usually much excitement at the beginning of a project or with the introduction of new concepts; however, if the design is customized poorly, teams and individuals will want to revert to the way the operation worked before. Ambler's (2014a) survey found that only 33% of participants rated adoption of agile software development as a success. Chikhale and Mansouri (2015) posited that organizations need to take additional steps and invest resources if they want to avoid resistance to change. As they bleakly noted, "many organizations adopt agile, and those that bring in outside help tend to adopt the methodology much faster than others, few can sustain their agile transformation" (p. 284).

Dikert et al. (2016) focused on the challenges and success factors for large-scale agile transformations. The scholars found that as large organizations attempt to scale agile methods, they frequently encounter issues because the design of most of these approaches is for use in small-to-medium sized companies. They also recommended that large organizations scale their

use of agile methods one unit at a time. Of course, this approach also creates issues, especially when integrating new approaches in contexts that are more used to the traditional waterfall method. Dikert et al. (2016) further argued that these types of situations highlight the need for more coordination across multiple teams that are working together within one system. One way in which large organizations can avoid these complications is ensuring the coordination of required documentation.

As supported by the *Agile Manifesto*, agile methodologies are founded, in part, on the assumption that minimal documentation is required to deliver change and product. As such, agile methods do not have a specified project management role assigned to the completion of project documents as the entire team performs this task. Research by Hoda and Murugesan (2016) identified some of the challenges that self-organizing teams face when using agile methods, including managing delayed and changing requirements, eliciting senior management sponsorship at the project level, handling missing or minimal acceptance criteria, and identifying dependencies at the task level.

Misra et al. (2009) found that when companies transition from a process-driven method to short, iterative cycles they are likely to face difficulties. As they note, agile methods are people-centric which differs greatly from the culture of many organizations. To fully implement agile methods, organizations need to also plan for a cultural change affecting both the team and business stakeholders. Both groups must adjust their attitudes and increase their knowledge of the agile method. Changing the culture of an organization can be difficult; however, it is possible when the process encompasses people at all levels and roles (Misra et al. 2009).

As the use of agile methodologies continues to grow, especially in large and global organizations, challenges will continue to occur, with few mitigation strategies. Both

researchers, Papatheocharous and Andreou (2014) and Khalil and Khalil (2016), found that introduction of more challenges and complexity happen with geographically located teams, and when projects become large-scale, these include a lack of knowledge and the resistance to change. Dikert et al. (2016) and Hoda and Murugesan (2016) also found that as global organizations use agile, there is a distribution of teams through many different locations (possibly across countries and time zones), which introduces additional challenges.

Many global companies encounter challenges when adopting agile methodologies. Adoption is difficult because of the need to coordinate cross-functionality and manage communication between teams; and integrate differing interpretations of agile methods, methodologies, and practices among teams and managers (Chikhale & Mansouri, 2015; Dikert et al., 2016; Goncalves & Lopes, 2014; Hoda & Murugesan, 2016; Misra et al., 2009). As is the case in smaller organizations, global companies must find ways to manage skepticism of a new way of working, build buy-in, and construct change coalitions. Lastly, global companies must find ways to provide training for those team members that lack experience at the individual level.

New approaches for facilitating the coordination of multiple teams and agile method practitioners have advanced new concepts. A couple of the new approaches are SAFe (Scaled agile framework), and R.A.G.E. SAFe is an interactive framework that manages agile projects on a large scale, and R.A.G.E., which stands for Recipes for Agile Governance in the Enterprise, provides guidance about how to make critical decisions (Goncalves & Lopes, 2014). Goncalves and Lopes further noted that with the introduction of new processes, there is increasing complexity; nevertheless, with the proper tools and training, problems can be identified early and mitigated sooner in the process.

Summary

There is a need for additional research on how global companies can implement Scrum methodology (Laanti & Abrahamsson, 2011; VersionOne, 2016a). Research continues to find it important to anticipate, recognize, and understand challenges common to the adoption, implementation, and scaling of agile software development strategies (Agile Alliance, 2016; Scrum Alliance, 2016c). Unfortunately, this creates a circular problem, as any time there is change, new challenges are also introduced (Dikert et al., 2016; Hoda & Murugesan, 2016; Khalil & Khalil, 2016; Papatheocharous & Andreou, 2014).

As discussed in the literature, there are many common challenges encountered by organizations and practitioners engaged in the implementation of the Scrum methodology in agile software development projects. Several studies found a lack of coordination among teams because individuals tend to assert autonomy and self-assignment (Dikert et al., 2016; Gandomani et al., 2014; Hoda & Murugesan, 2016). Papatheocharous and Andreou (2014), found Scrum teams and business stakeholders are resistant to change, which in turn limits knowledge and diminishes the capacity for understanding new methods and processes. Other scholars have identified delayed and changing requirements and the lack of executive support and sponsorship as the primary barriers (Dikert et al., 2016; Hoda & Murugesan, 2016). This dissertation focus is on identifying the CSFs needed to guide utilization of Scrum methodology successfully.

Critical Success Factors in Software Development Projects

CSF theory suggests that management should identify and focus on the few areas that must go right for the overall success of the project, a notion that applies to both organizations and projects (Bullen & Rockart, 1981; Daniel, 1961; Rockart, 1979). Initially, this theory was

applied by Bullen and Rockart (1981) to identify and measure organizational performance; however, as software development and projects have become core aspects of many organizations, application of CSF theory has expanded to include almost all areas of professional practice (Bullen & Rockart, 1981). This section addresses attributes of success in software development projects, the history of CSF theory and method, CSFs for the implementation of agile methodologies, Chow and Cao's (2008) research on project failure, and CSFs for software development projects.

Dimensions/Attributes of Success in Software Development Projects

There is an ongoing debate about what project success is and what helps to drive it. The TPM approach proposed in the *PMBOK® Guide* (PMI, 2013) supports the perspective that project charters document the criteria and requirements needed for project success. Criteria for success in TPM points towards the triple constraints of cost, scope and time, which are not easily measured and understood by all project participants (PMI, 2013). Nevertheless, Muller and Jugdev (2012) and Millhollan and Kaarst-Brown (2016) have stressed that project success attributes/criteria vary with each project due to a combination of influences ranging from personal, project, and team, to organizational success. As Muller and Jugdey and Millhollan and Kaarst-Brown have noted, conceptualizing and operationalizing project "success" is subjective.

The agile approach considers more than just the triple constraints of project success. Chow and Cao (2008) concluded there are five categories, "agile project success factors can be classified into five categories: organizational, people, process, technical and project" (p. 963). Along with these five categories of success factors, they identified four dimensions/attributes of success in agile software development projects, including quality, scope, timeliness, and cost. Muller and Jugdev (2012) further asserted that among the criteria of success most commonly

referenced in the project management literature are "(a) project efficiency, (b) impact on customers, (c) business success, and (d) strategic potential" (p. 764).

Overall, project success is ever-changing and based on the subjective perspectives of the stakeholders; therefore, there is no "one" definition for all scenarios. For example, Millhollan and Kaarst-Brown (2016) found that project stakeholders rarely agree on the definition of goals and measurements of success. As a result, Millhollan and Kaarst-Brown suggested a new focus on project outcomes and the process of project management. After reviewing 59 relevant articles, Millhollan and Kaarst-Brown concluded that the existing library of project management methodology has many tools capable of ensuring project success. Moreover, there is a need to alter the definition of project success. The Standish Group's "2015 Chaos Report" suggested one such definition—that the understanding of success is "on time, on budget and with a satisfactory result" (as cited in Hastie & Wojewoda, 2015).

Overview of the CSF Theory and Method

In 1961, Daniel wrote about the management information crisis. He noted that business managers received many reports and large amounts of data; however, they were not able to make decisions because they did not have the right information. Daniel (1961) posited that companies establish objectives and strategies before they begin to execute their plans and set control measures for performance. Then, he concluded, a company could adjust the control measures as needed to obtain the planned outcome.

Rockart (1979) introduced CSF theory and method as a way to build upon the work of Daniel (1961). After reviewing the by-product technique, the null approach, the key indicator system, and the total study process, Rockart (1979) argued that the CSF approach was effectively the best. He proposed that the CSF method would work in any industry, so long as there was a

consideration of the sources needed to identify the factors. Some of the sources include the structure of the industry; competitive strategy, industry position, and geographic location; environmental factors; and temporal factors.

Bullen and Rockart (1981) applied the CSF theory in multiple contexts, including the U.S. automotive and computer industries. They described CSFs as those areas that are necessary for a manager to reach the chosen goals successfully. Classification of CSFs along many different dimensions vary within the organization, including (a) the industry, (b) the corporation, (c) sub-units, (d) departments within the company, and (e) the individual. Other ways to classify CSFs are (a) internal versus external, and (b) monitoring versus building-adapting. They noted that the most important aspect of CSF theory is knowledge and identification of the factors that will help make change successful, then to understanding which of those factors to monitor.

CSF theory represents a shift in emphasis, as it is about limiting the perspective to the few areas where things must go right for the business to flourish. It also posits that if managers concentrate on the essential areas of activity (as opposed to all areas of activity), then it is possible to achieve favorable results and reach goals.

Critical Success Factors in Agile Software Development Projects

For this study, understanding CSF theory is important; however, it is equally critical to recognize how it has been applied to the technology industry, and more specifically, within the context of agile software development projects. As the field of IT emerged, management was initially not sure how to use it to their advantage to manage information and data. At the same time, Bullen and Rockart (1981) began applying CSF theory to different industries while shortly after that, Rockart and Crescenzi (1984) used the CSF to create frameworks that made it possible for management to see the uses for technology. To accomplish the application of the CSF

method, Rockart and Crescenzi recommended a three-phase approach, including (a) linking the systems, (b) utilizing the CSF interview technique and priorities, and (c) creating prototypes and reports to meet the needs of management. The emerging perspective was that understanding what causes failure in processes was just as important as understanding what creates success.

Chow and Cao's study on CSFs for agile software development projects. To identify possible CSFs in agile software development projects, Chow and Cao (2008) conducted a review of available research on project failure and project success. They found gaps in this body of knowledge, noting "success research cited in the literature [had been] mostly based on case studies or meta-data or compilations and observations of agile projects and practices" (p. 963). Through a review of extant literature, they identified success and failure factors based on lessons learned from projects identified in other studies, placing them into four different categories.

Building on previous research by Cohn and Ford (2003), Chow and Cao (2008) identified nineteen different failure factors and classified them into four different categories – organizational, people, process, and technical. These failure factors are the areas that must be mitigated and turned around to create success in agile software development projects. Failure factors, also known as challenges, are common in the research on agile methodology adoption (Brown, 2015; Chow & Cao, 2008).

Following the identification of the failure factors and challenges which had four dimensions and 19 factors, Chow and Cao (2008) reviewed and synthesized the literature on success factors. Building on previous research (Cohn & Ford, 2003; Lindvall et al., 2004), Chow and Cao (2008) identified 36 possible CSFs and classified them into five categories – organizational, people, process, technical, and project. Using reliability analysis and the Cronbach's alpha method, Chow and Cao (2008) narrowed down the initial list of 36 factors into

a list of 12 potential factors addressing four dimensions – Quality, Scope, Time, and Cost. Chow and Cao (2008) narrowed down the initial list of 36 factors into a list of 12 potential factors addressing four dimensions – Quality, Scope, Time, and Cost.

Chow and Cao proposed 12 hypotheses, each of which is numbered and paired with the four different dimensions of project success (quality, scope, timeliness, and cost), for a total of 48 hypotheses. Then, after collecting data, Chow and Cao ran multiple regression models to analyze which of the factors qualified as CSFs. Their findings supported 10 of the 48 hypotheses, which suggest a few of the CSFs have significance, including "a correct delivery strategy; a proper practice of agile software engineering techniques and; a high caliber team" (p. 969). However, as with any research, Chow and Cao (2008) encountered limitations, including unequal representation of all agile methodology, with a bias towards the XP; and minimal representation of US companies.

Brown's study of CSFs and agile projects. Building upon and replicating Chow and Cao's (2008) study, Brown (2015) conducted a quantitative, survey research design study using multiple regression analysis to further understand about the challenges of managing agile projects and the significance of each potential CSF. Research participants included a random sampling of agile users in various roles with a concentration of participants located in the US, mitigating one of the limitations encountered by Chow and Cao (2008). Participants in Chow and Cao's study were mainly familiar with XP method, and few reported having significant experience with other agile software development approaches.

Brown (2015) found that only "6 of the 12 factors were significant: project type, project schedule, project nature, management commitment, project definition process, and delivery strategy" (p. 103). Brown (2015) did not find 5 of the factors identified by Chow and Cao

(2008) to be significant, including team environment, team capability, customer involvement, project management process, and agile software engineering techniques. The difference in findings could be due to the background of participants, which was limited to US professionals. Brown (2015) also noted some limitations in his study, including a lack of focus on a specific organization type or size and limited representation of agile methodologies beyond XP.

Summary

As discussed in the literature, project success is about both the project and the stakeholders involved. For years, scholars (e.g., Chow & Cao, 2008; Hastie & Wojewoda, 2015; Millhollan and Kaarst-Brown, 2016; Muller & Jugdev, 2012) have been trying to determine the right formula for project success, and still, the definition is elusive. This study builds upon Chow and Cao's conceptualization and operationalization of project success, which is supported by their review of the literature (Cohn & Ford, 2003; Lech, 2013; Lindvall et al., 2004).

As discussed in this section, CSF theory argues that identifying the correct success factor will help drive the overall success of the project (Bullen & Rockart, 1981; Daniel, 1961; Rockart, 1979). While CSF theory has its origins in the works of both Daniel (1961) and Rockart (1979), it is Daniel's research that initially posited that managers should focus on key factors of success. The research conducted by Rockart (1979) built upon Daniel's CSF theory. To date, managers use CSF theory to obtain the information they need to be successful; however, CSF theory alone would not allow for the integration of IT and management.

As IT has expanded, companies were finding it increasingly clear that they needed approaches that were flexible and capable of quick adaptation. The creation of the *Agile Manifesto* in 2001 marked a major turning point. However, after a few years of using agile methodologies, it has become evident that more research is needed to connect CSF theory and

agile methodologies firmly. For instance, Chow and Cao (2008) advanced one of the only studies that link CSF theory and agile software development to identify CSFs for agile software development projects. This research is a promising beginning, but more research is needed.

This study builds upon the work of Chow and Cao (2008) and Brown (2015) for exploring and narrowing the limitations encountered by these previous studies. Specifically, this study is a major contribution because it focuses on Scrum methodology, as opposed to XP, which was the focus of the earlier research. Finally, this study focused on participants of large and distributed agile software development projects using Scrum methodology in U.S.-based global companies. Again, this contribution is important since the earlier research explored the context of smaller organizations, with only a few based in the US.

Summary

Chapter 2 serves as a synthesis of relevant literature and situates this study's research questions and theoretical framework within the context of the field of project management, the study of CSF theory, and the practice of agile methods. This chapter discusses the history of project management by explaining the TPM approach, and the view of it as a less-effective method. This chapter also introduces the emergence of the agile approach and the Scrum methodology in the field of software engineering and development. Finally, the chapter also discusses the foundations of CSF theory and method and its applicability to various contexts, including agile software development projects.

Due to the increased use of technology and the global expansion of companies, there is evidence that TPM is insufficient for the needs of many organizations, projects, and practitioners because it does not manage change effectively. In the current technological climate, it is critical

that businesses have the capacity to react quickly and efficiently to change. Agile software development methods and Scrum methodology have gained in popularity precisely because they address needs not being met by TPM approaches (Dikert et al., 2016; Hoda & Murugesan, 2016; Khalil & Khalil, 2016; Papatheocharous & Andreou, 2014). While the literature on TPM, including the *PMBOK® Guide* (PMI, 2013), recognizes the value offered by phased approaches and structure, many organizations have a greater need for flexible approaches that have the capacity to respond quickly to change.

Fernandez and Fernandez (2008) found that TPM and agile approaches to software development have many differences, and Misra et al. (2009) and Laanti and Abrahamsson (2011) advanced research on perceptions and ways to improve the transformation and adoption of agile software development methods. Sharp and Ryan (2011) took a different approach by focusing on the teams and how they adjust or work with being globally located. Each of these studies has supported the use of agile methods while also highlighting the challenges that companies encounter in their use of them. These challenges are why it is important that companies identify CSFs to ensure success.

Agile software development methodologies are people-centric and propose the need for minimal documentation, which allows for flexibility and quick delivery of the product. Among other agile software development methodologies, Scrum has become the most popular for companies and practitioners because of its flexibility and focus of quick product delivery (Laanti & Abrahamsson, 2011; VersionOne, 2016a). As organizations of all sizes, especially those large in scale and global in reach adopt, implement, and scale agile software development methodologies—including Scrum—they continue to face challenges. These problems include inadequate coordination and inefficient communication among distributed teams, lack of

knowledge (at both the team and management level), and the overall resistance to change held by project team members and stakeholders (Chikhale & Mansouri, 2015; Dikert et al., 2016; Goncalves & Lopes, 2014; Hoda & Murugesan, 2016; Kaleshovska et al., 2015; Khalil & Khalil, 2016; Matalonga et al., 2013; Papatheocharous & Andreou, 2014). These many challenges create an environment of confusion that can lead to unsuccessful projects; this is why the identification of CSFs is critical to assisting and guiding companies.

Given the challenges encountered by companies and practitioners in their adoption and scaling of agile methodologies for their software development projects, it has become important to understand the requirements for and predictors of success. Given the mixed results of previous research (e.g., Chow & Cao, 2008; Brown, 2015), and the need to focus on particular agile software development methodologies, this study is valuable and adds to the body of knowledge in the field of project management. By following a period of maturation of agile software development methodologies, this study furthered research examining the significance of CSFs. This study expanded upon the research model developed by Chow and Cao (2008) and later adapted by Brown (2015) while also addressing some of their limitations. The findings presented in this inquiry complicate previous research conducted by Chow and Cao, Brown, and other studies reviewed in this chapter.

CHAPTER 3. METHODOLOGY

Introduction

The purpose of this study was to examine the relationships between 12 independent variables (representing possible CSFs for agile software development projects) and the dependent variable of project success (consisting of four dimensions), as proposed by Chow and Cao (2008) and later adapted by Brown (2015). As proposed by Chow and Cao (2008), and later adapted by Brown (2015), there are 12 independent variables representing CSFs. The dependent variable, project success, consists of four dimensions – Quality, Scope, Time, and Cost. This study extends research conducted by Chow and Cao (2008) and Brown (2015) in numerous ways. First, it considers CSFs in agile software development projects, particularly those using Scrum methodology. This study also focused on organizations that are large, based in the US, and use distributed teams. This chapter addresses the study's methodology, including design and methods, population and sampling, setting, data collection, instrumentation, hypotheses, data analysis, validity and reliability, and ethical considerations.

Design and Methodology

A research approach is based on philosophical assumptions and established research methods, and entails the plans and procedures used to collect and analyze data (Creswell, 2014). The methodological approach and design of this study align with its research purpose and questions, which propose replicating and further examining the research model and hypotheses

proposed by Chow and Cao (2008), and later adapted by Brown (2015). This study examined the significance of 12 identified possible CSFs to understand the extent to which they contribute to the success of large and distributed agile software development projects that use Scrum methodology in U.S.-based global companies.

The study's research model, including variables, constructs, hypotheses, and survey questions, flow from Chow and Cao's (2008) model. Chow and Cao (2008) built the model on the concept, theory, and method of CSFs (Bullen & Rockart, 1981; Daniel, 1961; Rockart, 1979; Rockart & Crescenzi, 1984). The model used in this study understands project success as it is understood through project management literature (Cohn & Ford, 2003; Lech, 2013; Lindvall et al., 2004; PMI, 2013). CSFs are the few key areas where "things must go right for a business to flourish and attain a manager's goals" (Bullen & Rockart, 1981, p. 7). Similarly, the concept of project success consists of meeting the allotted time, cost, quality, scope or functionality (Cohn & Ford, 2003; Lech, 2013; Lindvall et al., 2004; PMI, 2013). Applying CSF theory to this study placed a focus on the significance of selected factors as contributors to the success of large and distributed agile software development projects using agile methods, particularly Scrum methodology.

An explanatory, quantitative, and survey research design served to measure the independent and dependent variables, test hypotheses, and answer research questions. First, an explanatory research design is appropriate when attempting to understand a phenomenon. Explanatory designs "attempt to explain the reasons for the phenomenon that the descriptive study only observed" (Cooper & Schindler, 2014, p. 23).

Second, this study engages with quantitative research paradigms informed by a post-positivist worldview. This perspective challenges "the traditional notion of absolute truth of

knowledge … when studying the behavior and actions of humans" (Creswell, 2014, p. 6). The post-positivist worldview proposes a structured approach to research and theory testing that begins with research questions to understand the relationships between variables. Collection of data occurs through instruments designed to capture a quantitative measure of participants' opinions and researchers' observations. In this study, testing of hypotheses was through the use of statistical procedures. These procedures facilitate evaluation and interpretation of the results while also proposing additional tests for further theory verification (Phillips & Burbules, as cited in Creswell, 2014).

Although considered, qualitative research methods were not appropriate for this study. The study examined the existence and significance of relationships between 12 possible CSFs for agile software development projects as well as four dimensions of project success. Since this research was interested in identifying the most significant variables (CSFs) as predictors for the dependent variable (each of four dimensions of project success), it was not a good fit for a qualitative paradigm that may have used open-ended interview questions or qualitative data analysis methods such as coding (Creswell, 2014).

Third, in this study, an online survey served to collect data from a sample of participants. The survey research design uses structured questionnaires to collect data and provide the "quantitative or numeric description of trends, attitudes, or opinions of a population by studying a sample of that population … with the intent of generalizing from a sample to a population" (Creswell, 2014, p. 13).

Chow and Cao's (2008) model constructs are psychometric variables measured with seven-point Likert-type scale survey questions. The collection of data in this study was from surveys completed by participants with one of the following roles: product owner, Scrum master,

software developer, business analyst, and/or tester for a large and distributed agile software development project. To be considered for the survey, the individual needed to be in a U.S.-based global company that uses Scrum methodology. The survey included questions regarding perceptions of the significance of CSFs and the resolution of projects.

Graphical and quantitative data analysis techniques, including multiple regression analysis, served to test the hypothesized relationships between various independent variables (possible CSFs) and the dependent variables (project success, consisting of four dimensions). Examining and measuring these relationships was one pathway for explaining which factors have the most positive impact on the success of large and distributed agile software development projects that use Scrum methodology.

Population and Sampling

The study's research questions proposed identifying relevant CSFs and measuring their significance as contributors to the success of large and distributed agile software development projects that use Scrum methodology for each of four success dimensions (Quality, Scope, Time, and Cost). For the study's goal to be accomplished, the inquiry engaged participants (including product owners, Scrum masters, software developers, business analysts, and/or testers in U.S.-based global companies) in quantifying their opinions regarding the significance of potential CSFs and measures of success for a chosen project. This research addressed some of the limitations Chow and Cao (2008) and Brown (2015) faced in their research, which included minimal participation by US professionals (for Chow and Cao) or participation by US professionals only (for Brown). Both of these previous studies also had a bias towards projects using XP and Feature-Driven Development methodologies. Since the publication of their

research, the Scrum methodology has become much more prevalent; therefore, it was important for this dissertation research to consider what changes, if any, would occur in their research model if there was a change in the agile methodology used.

Chow and Cao (2008) obtained responses from a sample of 408 participants. Brown (2015) received responses from a sample of 127 participants. This study had a target sample size of 150 participants. GPower* (g*power, 2016) served for calculating this sample size, using a .80 power, .05 error, and an effect size of .15. The result of this calculation was a minimum sample size of 127 participants; however, a larger sample size (150 participants) served to cover potential losses of sampling units' due to incomplete or nonsensical responses. The realized sample size was of 132 complete responses from 168 participants that responded to the survey.

Given the purpose of this research, its general population (Trochim, 2006) pertained to project managers and agile software development professionals in U.S.-based global companies. The criteria for participation in the survey included:

- active users of the Scrum methodology in the role of project manager, program
- manager, agile coach, product owner/manager, Scrum master, software developer, business analyst, quality assurance specialist, and/or tester;
- with a minimum of one-year professional experience;
- located around the world but working for a U.S.-based global company; and
- having participated in at least one completed large and distributed agile software development project using Scrum methodology for his/her current organization within the last year.

As part of sampling procedures (particularly, when providing informed consent), each participant was asked to confirm that he/she met the required criteria for participation in the study. Excluded individuals from this research did not have any experience using Scrum methodology.

A simple and random sampling strategy served to build the representative sample. Cooper and Schindler (2014) argued that simple random sampling "is the purest form of probability sampling with each population element having a known and equal chance of selection" (p. 349). Accordingly, the study/accessible population and sampling frame (Trochim, 2006) included members from the Scrum Alliance (n=53,918) and Scrum Study Community Group (n= 47,072) that met the criteria for participation outlined above. The accessible population and sampling frame were conducive to the focus and purpose of the study, while also being large enough to provide the needed number of qualified subjects.

As part of the sampling procedures, participants responded to an invitation to participate in the study. The message included a link to the survey (including the informed consent letter) and posted on the SCRUMstudy LinkedIn group page and the Scrum Alliance Facebook page. Members of these two groups include practitioners of the Scrum methodology that met criteria for participation in the study. The group administrators provided written permission for the recruitment of study participants through social media.

Setting

The current study was not conducted through, or in conjunction with, any organization, company, or sponsor. However, two organizations, the SCRUMstudy, and the Scrum Alliance agreed to serve as third-party resources, providing the access to participants as part of the study's sampling frame and strategy. The study setting was online via the World Wide Web (WWW).

The SCRUMstudy LinkedIn page and the Scrum Alliance Facebook page served to recruit participants, providing the link to the informed consent and survey. The SurveyMonkey tool was used to distribute the survey and informed consent, and to collect data.

Data Collection

All study data were from qualified individuals in U.S.-based global companies with the roles of the program manager, project manager, team member, agile coach, product owner/manager, Scrum master, software developer, business analyst, quality assurance specialist, and/or tester. Criteria for participation also included having participated in large and distributed agile software development projects using Scrum methodology within the last year.

Social science researchers commonly use online surveys for some reasons, including convenience, bias minimization, and options for multiple survey modes (Pew Research Center, 2017). For instance, Chow and Cao (2008) and Brown (2015) delivered surveys online via the WWW and collected a larger and broader sample of participants located in different geographical locations. Chow and Cao's web survey targeted members of user groups from the Agile Alliance organization. Brown took a different approach, using the Fluidsurveys online platform, which is no longer available and replaced by SurveyMonkey (Fluidsurveys, 2017). Fluidsurveys provided Brown with qualified participants as well as the software capabilities of survey creation, email distribution list management, survey distribution, and data collection and reporting.

For this study, all data were collected electronically through a web-based survey platform (SurveyMonkey) that provided a do it yourself (DIY) research survey tool. SurveyMonkey allows researchers to create, administer, analyze, store, and retrieve data from online surveys

(SurveyMonkey, 2017). The SurveyMonkey's Professional Gold plan, which students can purchase for $204 annually, allows researchers the capacity to deliver surveys with unlimited questions and responses, export data and generate reports, and complete text analysis (SurveyMonkey, 2017). Access to the SurveyMonkey platform and hosted surveys is password-protected, which ensures data confidentiality.

Data collection occurred in three phases, including (a) preparation, (b) data collection, and (c) post-data collection. First, many preparations happened before data collection. Capella University's IRB granted approval to conduct the study, recruit participants, and administer the survey to participants. Creation of the survey instrument was with DIY tools and features offered by the SurveyMonkey platform. Invitations to participate in the study were posted on the SCRUMstudy LinkedIn group page and the Scrum Alliance Facebook page. The invitation to participate in the study included a link to the online survey.

Second, during the data collection phase, participants responded to invitations to participate by following a hyperlink to the SurveyMonkey website, where they completed the study's online survey. The estimated completion time for the online survey was 20 minutes. The online survey instrument included three sections: (a) an informed consent with a description of the study and the survey; (b) instructions for completing the survey; and (c) survey questions. The informed consent served to acknowledge that participation in the study was voluntary. There were no incentives or rewards for participating in the survey given or offered to participants. Once a participant clicked "Yes" to acknowledge that he/she met the criteria for participation and understood the informed consent information, the next page provided instructions for completing the survey, consisting of 58 questions.

The online survey was configured to prevent the submission of incomplete answers. Participants had the option of returning to finish, as their incomplete survey was available for three weeks or until meeting the sample size of 150 participants, whichever came first. Posting of follow-ups occurred at 7 and 14 days following the initial invitation on the SCRUMstudy LinkedIn group page and the Scrum Alliance Facebook page. These follow-up postings served as reminders to potential participants about the study. The survey closed after 21 days of availability to participants. The SurveyMonkey platform served to store all data collected from participants.

Finally, during the post-data collection phase, a data file (SAV format) containing all participant responses, and downloaded from the SurveyMonkey platform, served to upload data into SPSS for data analysis. A password-protected computer and an external hard drive served to maintain the study data. This external hard drive will maintain the study data until destroyed, seven years following publication of the dissertation report.

Instrumentation

The survey instrument initially developed, validated, and used by Chow and Cao (2008), and then adapted and employed by Brown (2015), measured the study's constructs of CSFs and success in agile software development projects. Utilization of Chow and Cao's (2008) study instrument was without modifications. Chow, Cao, and Brown provided written permission to use/adapt this instrument for this study.

Chow and Cao's (2008) literature review on failure and success in agile software development was also essential to the methodology of this study. Chow and Cao first studied failure research and identified failure factors, or areas that are not a choice for the successful

resolution of software development projects. They also reviewed success research and identified success factors, which are related to the pursuit of the successful resolution of software development projects. They placed their failure and success factors into four categories, including (a) organizational, (b) people, (c) process and (d) technical. Success factors included one additional area: project. Chow and Cao further honed their list of failure and success factors to 12 possible CSFs for agile software development projects. These 12 independent variables divided into five categories include (a) organization, (b) people, (c) process, (d) technology, and (e) project.

Finally, Chow and Cao's (2008) study also examined project management research and identified attributes/criteria/measures of project success. They identified four dimensions of project success, including (a) quality, (b) scope, (c) time, and (d) cost. These dimensions correspond to commonly used criteria for assessing the resolution of projects as documented in project management literature (Cohn & Ford, 2003; Lech, 2013; Lindvall et al., 2004; PMI, 2013).

Survey Items

The study's survey instrument contains 58 questions. Table 1 reflects the breakdown of the survey by section number, section name, and question. Question 1 asks if the participant gives consent to participate (yes/no), and the rest of the survey consists of four different sections including participant demographics, project details, and dependent and independent variables.

Table 1

Details of Survey Instrument

Survey Section	Section Name	Survey Question Number
0	Consent to participate	#1
1	Participant demographic and project background information	#2-14
2	CSFs in agile software development project	#15-53
3	Perception of success of the agile software development project	#54-57
4	Additional comments	#58

Survey Scales

Consistent with Chow and Cao's (2008) study, the instrument in this study uses a seven-point Likert scale to measure the dependent and independent variables. These variables represent an individual's perceptions about CSFs in agile software development projects and project success attributes. Chow and Cao (2008) used seven-point Likert scales in order to "reflect the level or perception of success" (p. 965), and participants focused on one project to avoid ambiguity regarding success. To avoid ambiguity, each independent variable (representing a CSF in agile software development projects) includes seven prompts ranging from: "strongly disagree," to "strongly agree." Each of the four dimensions of the dependent variable (project success attributes) includes seven prompts ranging from "very unsuccessful," to "very successful."

The measures created in the original survey are an interval in nature for data analysis. Likert scales can be utilized for both ordinal and interval scale efficiently (Joshi, Kale, Chandel, & Pal, 2015). According to Allen and Seaman (2007) and Willits, Theodori, and Luloff (2016), the use of Likert scales in a survey instrument is common and an acceptable design format as it

provides a useful means for researchers to obtain data. The Likert scales in this study provide the same scale and range of responses for each item in the survey instrument. Data analysis in this study served to measure all variables at the interval/ratio level and obtain subscales from the survey, and then sum them to provide a total subscale score.

Hypotheses

This study examined the significance of CSFs for the success of large and distributed agile software development projects using Scrum methodology in U.S.-based global companies. Following on Chow and Cao (2008), the four research questions and hypotheses that guided the study are as follows:

Hypothesis One

RQ1: To what extent do Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) predict the quality of agile software development projects?

$H1_0$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) do not relate to the quality of agile software development projects.

$H1_a$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management

Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) significantly relate to the quality of agile software development projects.

Hypothesis Two

RQ2: To what extent do Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) predict the scope of agile software development projects?

$H2_0$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) do not relate to the scope of agile software development projects.

$H2_a$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) significantly relate to the scope of agile software development projects.

Hypothesis Three

RQ3: To what extent do Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software

Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) predict the time of agile software development projects?

$H3_0$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) do not relate to the time of agile software development projects.

$H3_a$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) significantly relate to the time of agile software development projects.

Hypothesis Four

RQ4: To what extent do Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) predict the cost of agile software development projects?

$H4_0$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) do not relate to the cost of agile software development projects.

$H4_a$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) significantly relate to the cost of agile software development projects.

Data Analysis

Given the explanatory nature of this study, graphical techniques (including histograms, probability-probability plots, and scatter plots) and quantitative data analysis techniques (descriptive statistics, multiple regression, and hypothesis tests) were used to examine the research model proposed by Chow and Cao (2008). Chow and Cao's research model hypothesizes the relationship between 12 independent variables (potential CSFs in agile software development projects) and one dependent variable (project success). The purpose of this approach is to learn "which factors can positively impact the success of an agile [software development] project[s]" (p. 965), and examine "the relative predictive importance of the independent variables" (p. 965).

The regression model yields an F test that helps analyze the relationship of agile CSFs and project success as a construct, as well as a sensitivity analysis of each CSF contribution to the variance of agile project success. For their data analysis, Chow and Cao (2008) employed two different regression models: full and optimized. For this study, SPSS Statistics software (version 24.0) served to conduct data exploration as well as calculate the descriptive statistics and full regression. First, descriptive statistical techniques shed light on the sample, including participant demographics and project size, length, location, etc. Second, graphical techniques

and multiple regression analyses served to understand (a) the distribution of the dependent variables (skewness and kurtosis tests), and (b) the importance and significance of each independent variable (full and optimized regression models and t-tests).

As discussed by Chow and Cao (2008), multiple regression analysis is appropriate given the study's intent to test hypothesized relationships between dependent and independent variables, as well as learn which of the CSFs have the most significant effect on the dimensions of the success of agile software development projects. As explained by Yale University (n.d.):

> Multiple linear regression attempts to model a relationship when there are two or more variables the formula given n observations is:
>
> Y = B0 + B1x1 + B2x2 + ……...+ Bkxk + E
>
> Where Y is the dependent variable, X1, X2, …, Xk are the independent variables, B1 is the regression coefficient, and E is the random error component. The value of the coefficient B1 determines the contribution of the independent variable X1, given that the other X variables are held constant, and B0 is the y-intercept. (para. 1-3)

According to Parke (2013), there are three assumptions about variables in multiple regression models. First, the independent variables are fixed, not random, so the measurement scale cannot change from one analysis to another. Second, the measurement of independent variables is without error. Lastly, there is a linear relationship between the independent and dependent variables. Moreover, as explained by Osborne and Waters (2002), multiple regression has other assumptions that need to be considered and tested. These assumptions include:

- Multivariate Normality, that is, variables have normal distributions.
- A linear relationship, that is, the relationship between the independent and dependent variables is linear.
- Reliability, that is, measures of variables are without error.

- Homoscedasticity that is, the variance around the regression line is the same for all values of the independent variable.

Osborne and Waters further stressed the need for monitoring these assumptions when evaluating data in multiple regression analysis.

Validity and Reliability

To evaluate the quality of quantitative research, researchers use the criteria of validity and reliability (Creswell, 2014). The following discussion addresses the consideration of these criteria for this study.

Validity

Validity seeks to answer whether one is measuring what he/she claims to be measuring. For instance, Roberts, Priest, and Traynor (2006) argued "reliability and validity are ways of demonstrating and communicating the rigor of research processes and the trustworthiness of research findings. If research is to be helpful, it should avoid misleading those who use it" (p. 41). According to Creswell (2014), "validity refers to whether one can draw meaningful and useful inferences from scores on specific instruments" (p. 250).

Similar to Chow and Cao's (2008) study, variables and survey constructs used in this study represent individual's perceptions regarding CSFs and attributes of success in agile software development projects that are measured using Likert scales. Survey items and scales used in this study are also consistent with those employed by Chow and Cao. In order to confirm the validity of their instrument, Chow and Cao used the Cronbach alpha and factor analysis techniques. They tested the survey instrument for content validity and readability by administering it to five individuals (representing the target participants in the study) and

gathering feedback to improve it. Chow and Cao's instrument was also adapted by Brown (2015), based on feedback from 37 participants in a field test. Both Chow and Cao (2008) and Brown (2015) adjusted the wording of questions to ensure clarity for the participant.

Reliability

As described by Mitchell and Jolley (2010), reliability entails ensuring the scores are consistent and stable for minimal or no influence by random error or chance. Creswell (2014) defined reliability as consistency in the test administration and scoring, allowing for internal consistency of the survey instrument.

Chow and Cao (2008) conducted reliability analysis on an initially compiled list of factors in order to synthesize a final list of 12 possible CSFs for agile software development projects. They used Cronbach's alpha method and two rounds of reliability analysis. Once they narrowed the list using these methods, they checked and reduced it further using a principal component factor analysis with Varimax rotation.

Ethical Considerations

This study engaged with and collected data from human subjects, specifically members of the SCRUMstudy and Scrum Alliance organizations. The requirement of ethical considerations was in order to build respect for human subjects. Capella University's IRB approved the study before starting with participant recruitment and data collection. Similarly, the survey design and all research activities followed principles and guidelines established by the *Belmont Report* (U.S. Department of Health & Human Services, 1979) regarding respect for persons, justice, and beneficence.

First, the *Belmont Report*'s principle of respect for persons requires that participants to have a choice about what does or does not happen to them (U.S. Department of Health & Human Services, 1979, part C). The information that participants reviewed in order to provide informed consent was part of the survey document. Informed consent ensures awareness, comprehension, and the choice to participate. The invitation to participate explained: (a) the purpose of the study; (b) type of data to be collected; (c) instructions for participating and expected time needed to complete the online survey; (d) risks entailed; (e) how anonymity would be guaranteed; and (f) who to contact with any questions.

Another aspect considered by the *Belmont Report*'s principle of respect for persons is the protection of subjects with diminished autonomy (U.S. Department of Health & Human Services, 1979). This research was not controversial, nor was it expected to cause harm or risk to any participant. Before proceeding with questions in the online survey, each participant had to review and provide informed consent. The informed consent form told the participant that:

- participation in the study was voluntary, and termination may be at any time without penalty or adverse consequences;
- there would be no connection of personal identities with responses to the survey;
- data were secure on a password-protected computer;
- reporting of data would be in aggregated format; and
- no incentive provided for participation in the study.

Second, the *Belmont Report*'s principle of justice requires that the participation level of any subject group correspond to the degree of benefit expected from the study (U.S. Department of Health & Human Services, 1979). The study's sampling frame included only those individuals working for a U.S.-based global company that had participated in a completed large

and distributed agile software development project using Scrum methodology within the last year. The sample frame was the group deemed to most likely benefit from the findings of the study.

Third, the *Belmont Report*'s principle of beneficence requires researchers avoid harming subjects and to benefit subjects instead when possible (U.S. Department of Health & Human Services, 1979). The content of the study's survey instrument was not expected to cause distress or harm to respondents. The study's data collection and analysis procedures included the steps and measures needed to ensure participants' privacy and confidentiality, including their demographic information. Findings of the study present demographic data about participants in consolidated format—not individually—which does not allow for the disclosure of any participant's identity.

Finally, the study's data collection and analysis procedures addressed the consideration of security of data. During data collection, the SurveyMonkey platform served to store data collected from participants. Following completion of data collection (and after downloading survey data from SurveyMonkey), a password-protected computer and an external hard drive served to maintain the study data. This external hard drive will maintain the study data until destroyed: seven years following publication of the dissertation report.

CHAPTER 4. RESULTS

Introduction

The purpose of this explanatory, quantitative, and survey study was to examine the relationships between 12 independent variables (representing possible CSFs for agile software development projects) and the dependent variable of project success (consisting of four dimensions), as proposed by Chow and Cao (2008) and later adapted by Brown (2015). Given the popularity and growth in adoption of the Scrum methodology for agile software development projects, building upon Chow and Cao's study and examining its research model with a focus on large and distributed projects using Scrum methodology in U.S.-based global companies is valuable and contributes to the body of knowledge in the field of project management.

Participants in the study included active users of the Scrum methodology that had participated in at least one completed large and distributed agile software development project using Scrum methodology for a U.S.-based global company within the last year. The characterization of an active user of the Scrum methodology included practitioners in the role of project manager, program manager, agile coach, product owner/manager, Scrum master, software developer, business analyst, quality assurance specialist, and/or tester with at least one year of experience.

A survey consisting of 58 questions, and originally designed and used by Chow and Cao (2008) served to collect data from participants in this study. The first question of the survey

asked participants to accept or reject participation in the study. The remaining 57 questions were presented in four sections:

- Section I – Demographics (questions 2-14): Free-form text questions to capture data about participants' demographics.
- Section II – Success Factors of the Agile Project (questions 15-53): Questions using a seven-point Likert scale to capture participants' perceptions regarding potential CSFs for agile software development projects.
- Section III – Perception of Success of the Agile Project (questions 54-57): Questions using a seven-point Likert scale to capture participants' perceptions regarding four criteria of success for agile software development projects.
- Section IV – Additional Comments (question 58): An open field for participants to add comments and/or any information not previously covered by the other questions.

Although there were no modifications to the survey developed by Chow and Cao (2008), there was a minor change to data collection procedures as described in chapter 3. The study was available to participants for a total of 29 days, eight days longer than the anticipated maximum, to allow for sufficient responses. After the first posting, seven additional posts occurred in the social media groups – each week on Monday and Thursday after the initial post, serving to promote participation in the survey.

Modification to the proposed process happened because completed surveys were slower to be received than anticipated. In the attempt to obtain more participation, the request for participants was placed more frequently and over a longer period. The first response request was posted on both sites June 19, 2017, and then twice a week over the next four weeks until the last posting on July 17, 2017.

This chapter serves to provide a detailed account of the data collection and analysis activities and a discussion of the findings of the study. The first section, data collection results, presents a review the data collection approach and results. The second section, descriptive analysis, presents the analysis of the data in detail along with discussing the results for the demographics, linearity, and significance of each dimension and factor. The third section, analysis of hypotheses, presents the analysis and review of the results and evaluation of hypotheses. The last section, summary, serves to summarize the overall performance of the study.

Data Collection Results

The sample of the study was representative of the population of practitioners of the Scrum methodology in U.S.-based global companies. These practitioners included team members (product owners, Scrum masters, software developers, business analysts, and/or testers) of large and distributed agile software development projects using Scrum methodology completed within the last year in U.S.-based global companies. The sample frame included 53,918 members of the SCRUMstudy LinkedIn group page (SCRUMstudy, 2015) and 47,072 members of the Scrum Alliance Facebook page (Scrum Alliance Facebook, 2017). A simple random sampling strategy was used to recruit 150 participants that responded to an invitation to participate in the study and completed the survey.

Invitations to participate in the study posted online on the SCRUMstudy LinkedIn group page (SCRUMstudy, 2015) and the Scrum Alliance Facebook page (Scrum Alliance Facebook, 2017), included a hyperlink that took participants to the study's survey available online in the SurveyMonkey platform (SurveyMonkey, 2017). The online survey included the informed

consent form, which before presenting survey questions, served to ask participants to confirm that they met the criteria for participation and agreed to the terms of participation in the study.

Over the five-week period during which the study's survey was available online, a total of 168 participants responded, including 132 complete responses (79%) considered for data analysis, and 36 incomplete responses (21%) not considered for data analysis. SurveyMonkey provided the means for downloading data into an SAV format file for use in SPSS for data analysis.

The study's sample size was smaller than the 408 participants in Chow and Cao's (2008) study, but larger than the 127 participants in Brown's (2015) study. Nevertheless, this sample size was larger than the minimum recommended sample size of 127 participants – calculated using GPower* (g*power, 2016), and using a .80 power, .05 error, and an effect size of .15.

A few limitations of the data collection procedures and data collected are worth noting. First, the study's sample size was relatively small at 132 participants compared to the overall agile community population consisting of thousands of potential participants on both sites, SCRUMstudy and Agile Alliance, used to obtain participants. However, the 132 participants were sufficient to meet the required sample size. Second, questionnaires do not capture the emotions and feelings of the participant, while rankings are subjective and based on participants' recollection and opinions regarding a project they completed in the past (libweb, n.d.). Third, participants did not represent any one organization or industry; they were from many different organizations and industries suggesting that results may be different if the evaluation was on only one organization or industry. Fourth, the study survey did not provide participants with a clear definition or criteria for describing a large agile software development project, therefore the interpretation of the meaning of large project was subject to each individual's perception.

Finally, the instrument could have better-captured participant demographics with a few changes to the questions along with having multiple choice answers, which do not exist in the current device. These changes would have allowed for better understanding the results, including if processes varied based on overall demographics.

Descriptive Analysis

The study's survey includes 13 questions about the agile software development project, the business setting, and the role and experience of participants. Demographic questions asked participants to provide the project profile, which served to characterize the type of agile software development projects represented by the study's findings. The following discussion describes projects represented in the survey results, including methodology, project length and location, company size, team size, and participant role and experience.

Agile Software Development Methodology Used in Projects

All the projects considered by participants in the study met the criteria of having used the Scrum methodology, even though some showcased a combination of Scrum and Kanban (Scrumban) or an added program management layer of SAFe to the overall Scrum methodology. As shown in Table 2, over 50% of participants reported that their project used the Scrum methodology. Another 39.4% of participants indicated that their project used a combination of the Scrum methodology with the overlay of the program structure of SAFe. The remaining 9.1% of participants reported that their project applied the Scrum methodology while also utilizing the Kanban prioritization process of a continuous cycle.

Table 2

Project Profile – Agile Method

Method	Frequency	Percent	Valid Percent	Cumulative Percent
SAFe	52	39.4	39.4	39.4
Scrum	68	51.5	51.5	90.9
Scrumban	12	9.1	9.1	100
Total	132	100	100	

Note: Distribution of agile methodologies used for project process.

Length of Projects

The study's survey asked participants to provide their perceptions regarding potential CSFs and success criteria for a completed large and distributed agile software development project using Scrum methodology within their current organization and completed within the last year. As shown in Table 3, most projects (60.6%) lasted over a year with 27.3% of them lasting over 25 months.

Table 3

Project Profile – Length of Project in Months

Months	Frequency	Percent	Valid Percent	Cumulative Percent
1-6	17	12.9	12.9	12.9
7-9	30	22.7	22.7	35.6
10-12	5	3.8	3.8	39.4
13-18	23	17.4	17.4	56.8
19-24	21	15.9	15.9	72.7
Over 25	36	27.3	27.3	100
Total	132	100	100	

Location of Projects

Agile software development projects considered by the study's participants were executed by a U.S.-based global company and within the context of distributed teams. As shown in Table 4, of the projects represented in the study, 65.2% were in the US, while 34.8 percent were global.

Table 4

Project Profile – Location of Project

Location	Frequency	Percent	Valid Percent	Cumulative Percent
Global	46	34.8	34.8	34.8
U.S.	86	65.2	65.2	100
Total	132	100	100	

Note: Location represents the location of the participant. Global represents the participant was outside of the U.S.

Company Size

The study's survey asked participants about the size of the company that executed the represented project. As illustrated in Table 5, most of the companies had over 1000 employees. This question could have captured more information had there been more detailed ranges.

Table 5

Project Profile – Company Size (Number of Employees)

Number of Employees	Frequency	Percent	Valid Percent	Cumulative Percent
1-100	3	2.3	2.3	2.3
101-1000	6	4.5	4.5	6.8
1000 or more	123	93.2	93.2	100
Total	132	100	100	

Size of Project Teams

The study's survey asked participants to identify the size of the represented project. Considering the size of a team serves to describe the size and scope of projects represented in the study. As shown in Table 6, a total of just over 48% of the projects were conducted using less than 20 team members.

Table 6

Project Profile – Number of Team Members on the project

Number of Team Members	Frequency	Percent	Valid Percent	Cumulative Percent
Less than 10	30	22.7	22.7	22.7
11-19	34	25.8	25.8	48.5
20-29	15	11.4	11.4	59.8
30-39	11	8.3	8.3	68.2
40-49	12	9.1	9.1	77.3
50-100	18	13.6	13.6	90.9
101-300	12	9.1	9.1	100
Total	132	100	100	

Roles of Participants in Projects

Criteria for participation in the study included being a practitioner of the Scrum methodology in a U.S.-based global company. These practitioners represented roles such as product owners, Scrum masters, software developers, business analysts, and testers.

The study's survey asked participants to identify their role in the evaluated projects. This data is important because different positions may have different opinions about what counts as success. As shown in Table 7, the Scrum master role was the most common position held by participants at 27.3%. The next highest was the developer role with 19.7%.

Table 7

Respondent Profile – Job Role

Role	Frequency	Percent	Valid Percent	Cumulative Percent
Agile Coach	2	1.5	1.5	1.5
Business Analyst	15	11.4	11.4	12.9
Developer	26	19.7	19.7	32.6
Manager	17	12.9	12.9	45.5
Product Owner	8	6.1	6.1	51.5
Program Management	3	2.3	2.3	53.8
Project Manager	11	8.3	8.3	62.1
Scrum Master	36	27.3	27.3	89.4
Team Member	13	9.8	9.8	99.2
Tech lead	1	0.8	0.8	100
Total	132	100	100	

Participants' Experience Using Scrum Methodology

The study's survey asked participants to identify their years of experience using Scrum methodology. A participant's experience level provides a knowledge level necessary to understand the overall process structure. As shown in Table 8, a majority (95.9%) of participants have over two years of experience. Similarly, as shown in Table 9, 60.6% of respondents have participated in 1-5 projects using Scrum methodology.

Table 8

Respondent Profile – Years of Experience with Agile Projects

Length of Experience	Frequency	Percent	Valid Percent	Cumulative Percent
Over 1 year	8	6.1	6.1	6.1
2-5 years	67	50.8	50.8	56.8
6-10 years	34	25.8	25.8	82.6
11+ years	23	17.4	17.4	100
Total	132	100	100	

Table 9

Respondent Profile – Number of Agile Scrum Projects Involved with

Number of Projects	Frequency	Percent	Valid Percent	Cumulative Percent
1-5	80	60.6	60.6	60.6
6-10	14	10.6	10.6	71.2
More than 10	38	28.8	28.8	100
Total	132	100	100	

In summary, based on the results of the study, most large companies utilize Scrum methodology's regular practices, with most of the projects lasting over a year. The project teams are typically less than 20 members who have over two years' experience and have worked on one to five projects.

Analysis of Hypotheses

This section outlines the research model, variable and survey items, and describes the analysis procedures and the results for each of the dependent variables using the multiple regression model.

Research Model, Variables, and Survey Items

The 12 independent variables in the study's research model (Chow & Cao, 2008) represent five categories of potential factors impacting the success of agile software development projects. Project success, conceptualized as four independent dimensions – Quality, Scope, Time, and Cost, is the dependent variable in the study's research model. Expanding on Chow and Cao's research model, the design of this study included null and alternative hypotheses for

the relationship between 12 CSFs and each of four dimension of agile project success, for a total of 4 sets of null and alternative hypotheses, one for each of four research question.

An instrument initially developed by Chow and Cao (2008) and delivered using the SurveyMonkey service, served to measure the study's dependent and independent variables. The study's survey included 43 items measuring 16 constructs. Participants answered each of these survey items using a seven-point Likert scale varying from "strongly disagree" to "strongly agree." The "N/A – Not applicable / don't know" option was also available to allow for those areas that did not apply to the project or was not understood by the participant. In Table 10 the abbreviation CSF represents the critical success factor along with assigning a number to each of the 12 factors. The 12 factors are within a category.

Data Analysis Procedures

The study resulted in 132 completed surveys representing the opinions of practitioners of the Scrum methodology regarding potential factors impacting the success of large and distributed agile software development projects. The IBM SPSS Statistics software, version 24.0, served to complete data exploration, including a graphical representation of data, descriptive statistics, and multiple regression.

As discussed earlier, descriptive statistics helped to provide frequencies and percentages and describe the sample of participants in the study. It was important to begin the review by describing the sample and understanding the demographics of three different areas: the project, the organization, and the participant.

Table 10

Survey Items – Abbreviations and Prompts

Abbreviation	Prompt Questions
CSF Category - Organizational Factors/ Dimension	
CSF1 - Management Commitment	15,16
CSF2 - Organizational Environment	17,18,19, 20, 22,45
CSF3 - Team Environment	21, 27,50,51
CSF Category - People Factors/ Dimension	
CSF4 - Team Capability	23,24,25,26,32,46
CSF5 - Customer Involvement	28,36,37
CSF Category - Process Factors/ Dimension	
CSF6 - Project Management Process	30,31,32,33,34,35
CSF7 - Project Definition Process	29,52,53
CSF Category - Technical Factors/ Dimension	
CSF8 - Agile Software Techniques	38,39,40,41,42
CSF9 - Delivery Strategy	43,44
CSF Category - Project Factors/ Dimension	
CSF10 - Project Nature	47
CSF11 - Project Type	48
CSF12 - Project Schedule	49
Perceived Success Attributes/ Dimension	Dependent Variable
Y1 – Quality	54
Y2 – Scope	55
Y3 – Time	56
Y4 – Cost	57

Note: CSFs 1-12 are independent variables and the Perceived Success Attributes/ Dimensions are dependent variables.

Graphical techniques served to summarize the data in a diagrammatic way. These techniques included histograms, probability-probability (P-P) plots, scatter plots. Similarly, quantitative methods used in the study included (a) descriptive statistical techniques for describing the sample, including participant demographics and project size, length, location, etc.; (b) skewness and kurtosis tests for examining the distribution normality of the dependent

variables, and (c) multiple regression analysis and t-test for measuring the relationship between the independent and dependent variables and the importance and significance of each independent variable. As an initial step, procedures proposed by Osborne and Waters (2002) helped to test the assumptions of multiple regression. Following is a discussion on these tests and their results.

Testing assumptions of the multiple regression models. When using multiple regression, it is important to meet certain assumptions. The assumptions include normality, or normal data distribution for modeled variables; and linearity, or a linear relationship between the dependent and independent variables. Also assumed is the reliability of measures for modeled variables; and homoscedasticity, or consistent variance of errors across all levels of the independent variable (Osborne & Waters, 2002).

Chow and Cao (2008) tested the validity and reliability of the survey items, and because of the use of the same survey in this study, there was no further testing conducted in these areas. Full multiple regression histograms, P–P plots, and scatter plot served to test and evaluate the other assumptions, which represent meeting the assumptions for all modeled variables in the study. Charts and graphs of the full regression models are displayed.

Full multiple regression model. Full multiple regression analyses served to test hypotheses for the study's research model, that is, examine the significance of the contribution of the predictor (independent) variables to the outcome of the dependent variable. First, the use of four different regression models occurred because there were four dimensions of project success with each serving as a dependent variable. In the full multiple regression, each model used all 12 independent variables for testing. The models had the multiple correlation coefficients (R), and the coefficient of determination (R^2) calculated. Each independent variable had the coefficients

B and β along with the t-value calculated. Only those variables with a positive B and β relationship were considered a CSF. Also, reviewed was the significance level for each independent variable and the distribution of normality of the dependent variables. Only the variables with a significance level (p) equal to or greater than .1 for the full model were considered CSFs.

Hypotheses

Hypothesis One Testing Results

$H1_0$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) do not relate to the quality of agile software development projects.

$H1_a$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) significantly relate to the quality of agile software development projects.

The following tables and figures summarize the results of data analysis for hypothesis one. Table 11 reflects the descriptive statistics for each of the variables reflecting the mean and standard deviation. Hypothesis one was tested using linear regression and found that the Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process,

Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, Project Schedule) were significantly related to the quality of agile software development projects (Table 13 – ANOVA; alpha = 0.05; sig. < 0.000). Thus, the null hypothesis ($H1_0$) was rejected, and the alternate hypothesis ($H1_a$) accepted. An effect size analysis found a medium relationship between Scrum CSFs and Quality (Gloeckner, Gliner, Tochterman, & Morgan, 2001, p. 227; Table 12 – Model Summary; Adjusted R Square = 0.397). A visual inspection of residuals (Figures 3, 4, and 5) in histogram, P-P plot and scatter plot found no violations of the normality or linearity assumptions.

Table 11

Descriptive Statistics of Scrum CSFs and Quality

Descriptive Statistics

	Mean	Std. Deviation	N
Project Quality	5.5682	1.12700	132
Management Commitment	5.4697	1.82723	132
Organization Environment	4.7285	1.25677	132
Team Environment	4.4905	1.53129	132
Team Capability	5.4985	1.04567	132
Customer involvement	5.2348	1.39959	132
Project Management Process	5.1553	1.22148	132
Project Definition Process	4.9773	1.41013	132
Agile Software Engineering Techniques	4.7939	1.24691	132
Delivery Strategy	5.4015	1.15982	132
Project Nature	6.1212	1.25419	132
Project Type	5.4848	1.35063	132
Project Schedule	5.5758	1.19873	132

Table 12

Model summary of Scrum CSFs and Quality

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.672[a]	.452	.397	.87532

a. Predictors: (Constant), Project Schedule, Delivery Strategy, Project Definition Process, Project Type, Management Commitment, Customer involvement, Project Nature, Organization Environment, Agile Software Engineering Techniques, Team Capability, Team Environment, Project Management Process
b. Dependent Variable: Project Quality

Table 13

ANOVA of Scrum CSFs and Quality

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	75.210	12	6.268	8.180	.000[b]
	Residual	91.176	119	.766		
	Total	166.386	131			

a. Dependent Variable: Project Quality
b. Predictors: (Constant), Project Schedule, Delivery Strategy, Project Definition Process, Project Type, Management Commitment, Customer involvement, Project Nature, Organization Environment, Agile Software Engineering Techniques, Team Capability, Team Environment, Project Management Process

Figure 3 shows the histogram of the residual in the regression model for the Y1 dependent variable, which suggests a normal distribution. Figure 4 shows the P-P plot for the residual of the regression model for the Y1 dependent variable, which is approximately a straight

line. Figure 5 is a scatter plot reflecting the linearity as well. These tests suggest that the distribution of the Y1 dependent variable is almost normal and there are no violations of the normality or linearity assumptions.

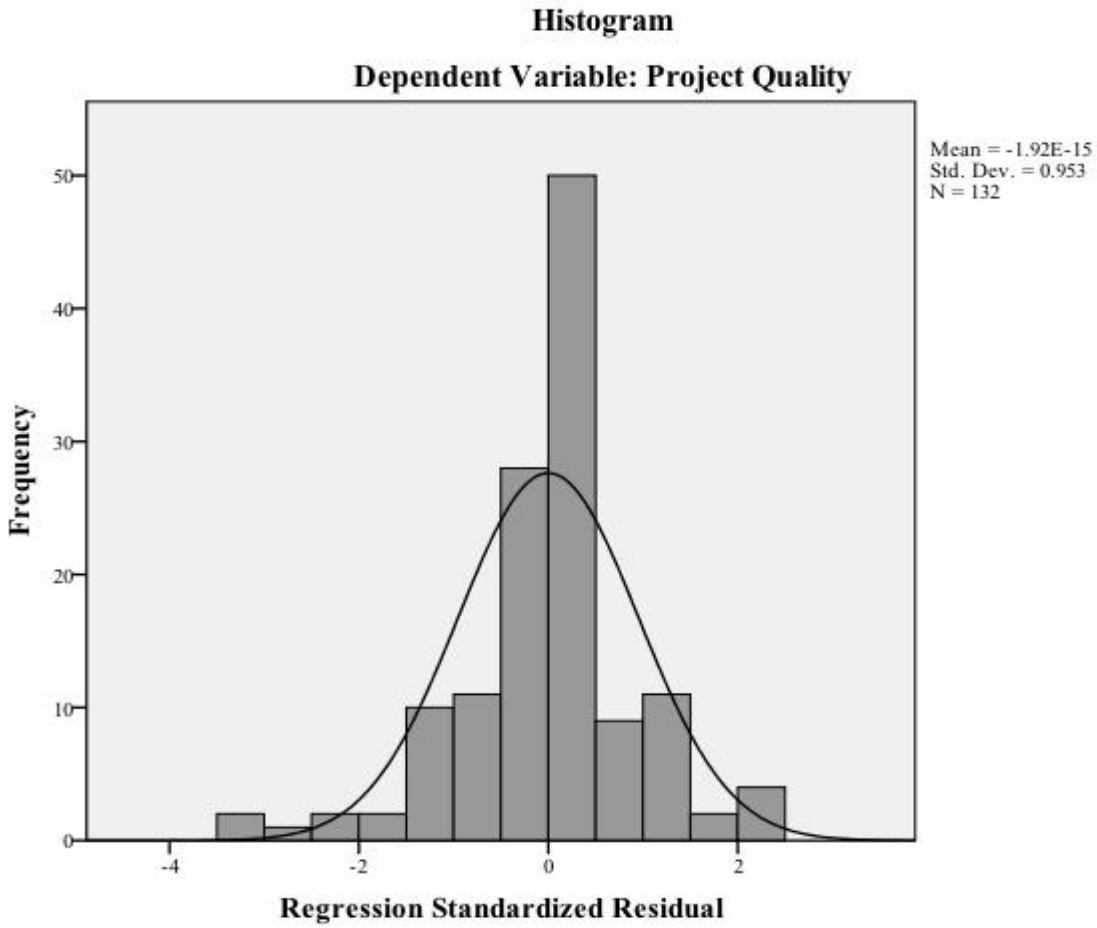

Figure 3. Histogram of the Regression Standardized Residual Model for Y1 (Quality)

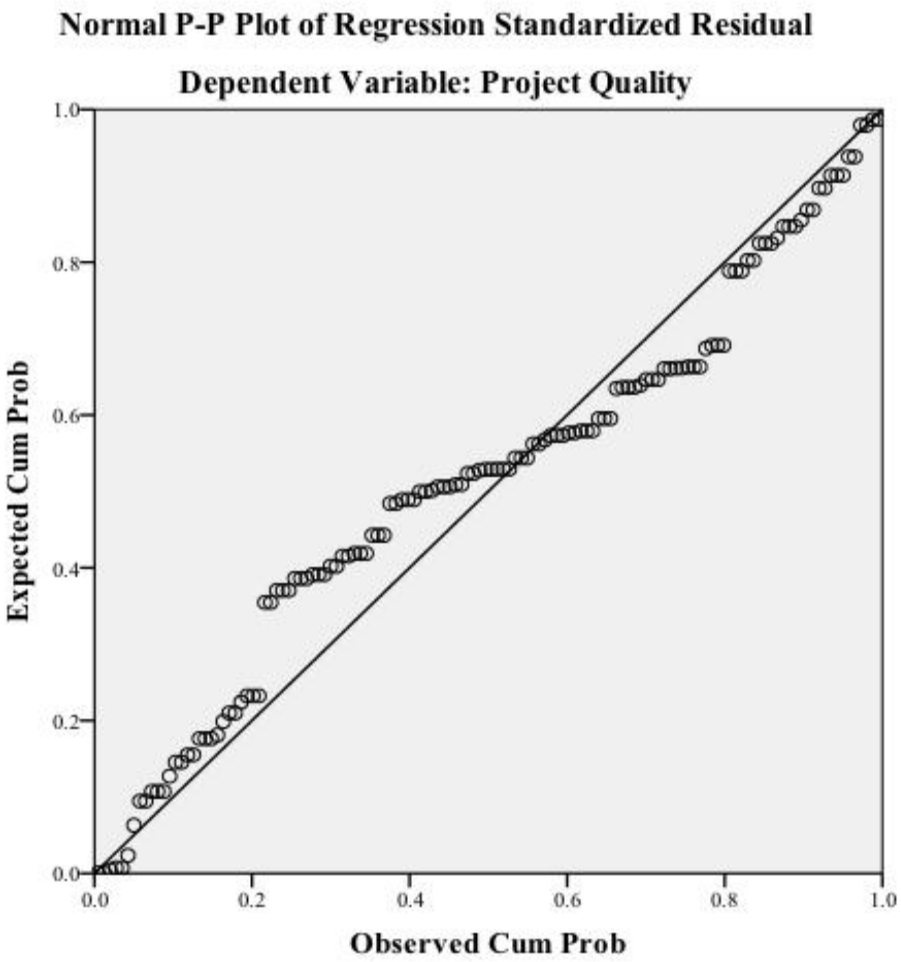

Figure 4. Normal P-P Plot of Regression Standardized Residual for Dependent Variable Y1 (Quality)

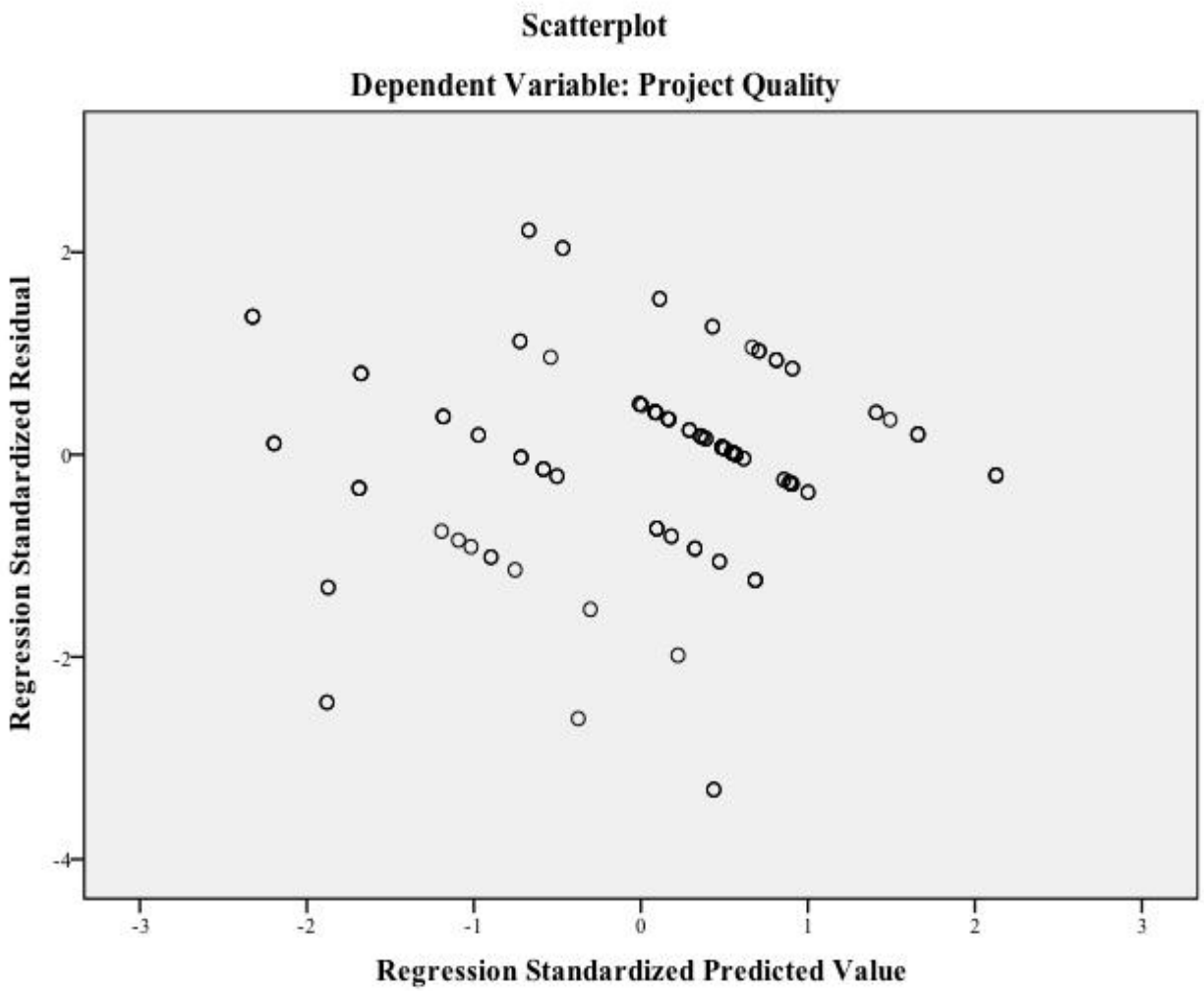

Figure 5. Normal Scatterplot of Regression Standardized Residual for Dependent Variable Y1 (Quality)

Table 12 shows the R and R^2 values for the full multiple regression analysis of the Y1 dependent variable. The value of the correlation coefficient R of .67 suggests a high correlation between variables in the model. R^2 is the coefficient of determination and shows that the model explains 45.2% of the variance.

Table 14 shows the details of the coefficients for all 12 independent variables. The table shows both the unstandardized B coefficient and standardized B coefficients. The unstandardized coefficients explain the contribution the variable made towards the prediction of

Y1 when all other variables were held constant. The standardized coefficients describe the extent of each predictor variable's contribution to the joint prediction of Y1. The table also shows the probability of sample occurrence in testing the null hypothesis as reflected by the t-test. Four variables showed relative higher B and β values, a higher absolute t-value, along with a significance value at below .1, including:

- Team Capability (B = .5, β = .464, t = 3.386),
- Delivery Strategy (B = .247, β = .254, t = 2.558),
- Project Type (B = -.21, β = -.251, t = -2.992), and
- Organization Environment (B= -.194, β= -.217, t= -1.757).

The variables Project Type and Organizational Environment showed negative B and β values and a negative relationship with the Y1 dependent variable. These results suggest that the less the scope definition and flexibility (Project Type) and/or not having the proper environment (Organizational Environment) contribute to the reduction of the Quality dimension of project success.

Table 14

Coefficients of Scrum CSFs and Quality

Coefficients[a]

Model		Unstandardized Coefficients B	Unstandardized Coefficients Std. Error	Standardized Coefficients Beta	t	Sig.
1	(Constant)	2.856	.646		4.424	.000
	Management Commitment	-.061	.059	-.099	-1.042	.300
	Organization Environment	-.194	.111	-.217	-1.757	.082
	Team Environment	.123	.099	.167	1.234	.220
	Team Capability	.500	.148	.464	3.386	.001
	Customer involvement	-.078	.081	-.097	-.970	.334
	Project Management Process	-.043	.138	-.047	-.313	.755
	Project Definition Process	-.002	.067	-.003	-.035	.972
	Agile Software Engineering Techniques	.197	.125	.218	1.581	.117
	Delivery Strategy	.247	.097	.254	2.558	.012
	Project Nature	.022	.102	.024	.214	.831
	Project Type	-.210	.070	-.251	-2.992	.003
	Project Schedule	.009	.075	.009	.117	.907

a. Dependent Variable: Project Quality

The analysis of these results suggests two out of the 12 CSFs for the Y1 dependent variable, Team Capability, and Delivery Strategy were significant enough to be considered CSFs that contribute to the Quality dimension of project success. The relationship between the factors and dependent variables can be summarized as follows:

Project Success (Quality) = f (Team Capability, Delivery Strategy)

$R^2 = .45$

Team Capability: $\beta = .464$, $t = 3.386$

Delivery Strategy: $\beta = .247$, $t = 2.558$

Hypothesis Two Testing Results

$H2_0$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) do not relate to the scope of agile software development projects.

$H2_a$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) significantly relate to the scope of agile software development projects.

The tables and figures below summarize the results of data analysis for hypothesis two. Hypothesis was tested using linear regression and found that the Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, Project Schedule) were significantly related to the scope of agile software development projects (Table 16 – ANOVA; alpha = 0.05; sig. < 0.000). Thus, the null hypothesis ($H2_0$) is rejected, and the alternate hypothesis ($H2_a$) accepted. An effect size analysis found a medium to strong relationship between Scrum CSFs and Quality (Gloeckner et al., 2001, p. 227; Table 15 – Model Summary; Adjusted R Square = 0.450). A visual inspection of residuals (Figures 6, 7, and 8) in histogram, P-P plot and scatter plot found there is no violation of the normality or linearity assumptions.

Table 15

Model summary of Scrum CSFs and Scope

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.707[a]	.500	.450	.85177

a. Predictors: (Constant), Project Schedule, Delivery Strategy, Project Definition Process, Project Type, Management Commitment, Customer involvement, Project Nature, Organization Environment, Agile Software Engineering Techniques, Team Capability, Team Environment, Project Management Process
b. Dependent Variable: Project Scope

Table 16

ANOVA of Scrum CSFs and Scope

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	86.475	12	7.206	9.933	.000[b]
	Residual	86.335	119	.726		
	Total	172.811	131			

a. Dependent Variable: Project Scope
b. Predictors: (Constant), Project Schedule, Delivery Strategy, Project Definition Process, Project Type, Management Commitment, Customer involvement, Project Nature, Organization Environment, Agile Software Engineering Techniques, Team Capability, Team Environment, Project Management Process

Figure 6 shows the histogram of the residual in the regression model for the Y2 dependent variable, which suggests a normal distribution. Figure 7 shows the P-P plot for the residual of the regression model for the Y2 dependent variable, which is approximately a straight line. Figure 8 shows the scatterplot of the Y2 variable. These tests suggest that the distribution

of the Y2 dependent variable is almost normal and there are no violations of the normality or linearity assumptions.

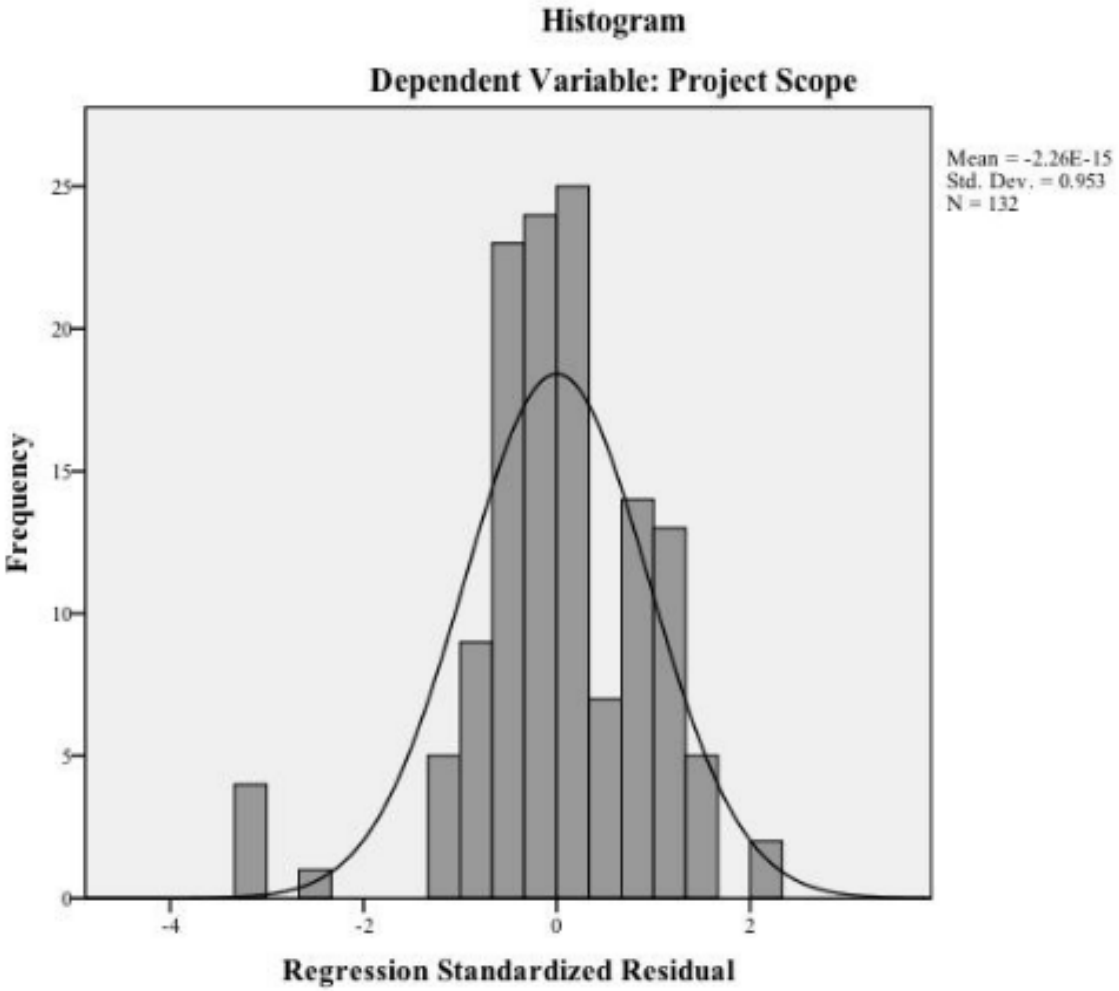

Figure 6. Histogram of the Regression Standardized Residual Model for Y2 (Scope)

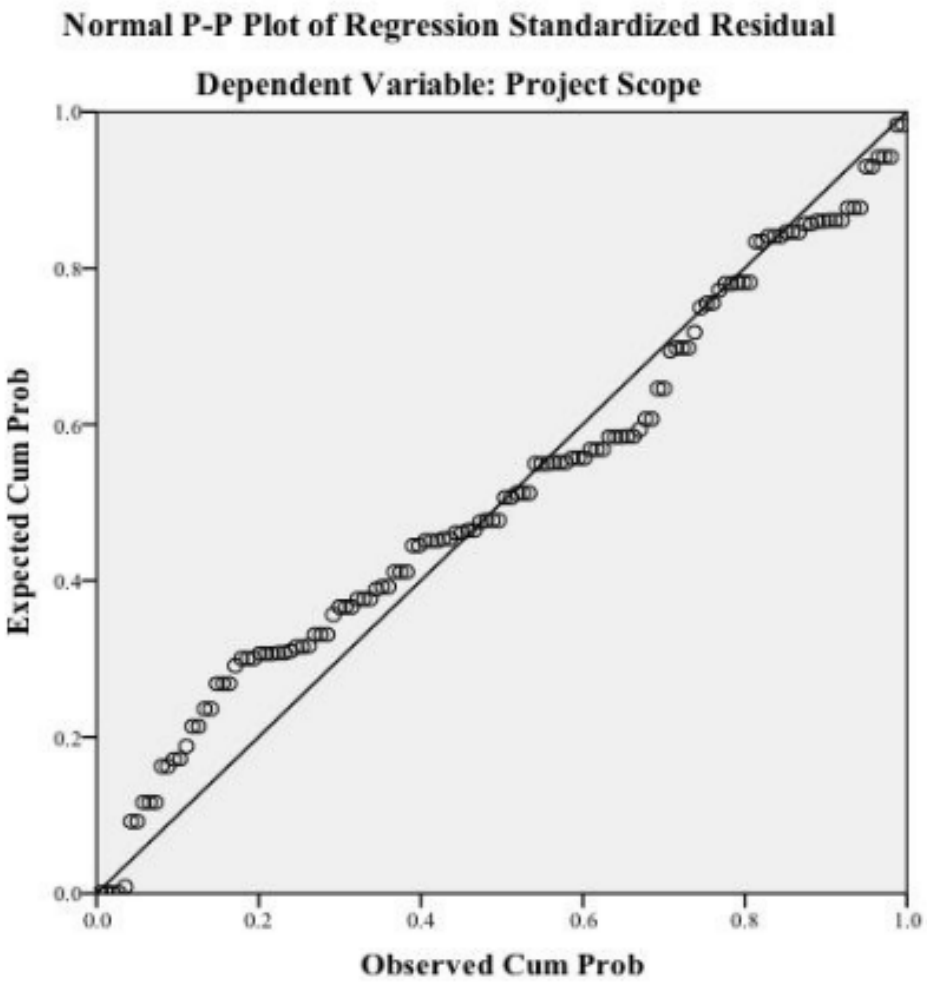

Figure 7. Normal P-P Plot of Regression Standardized Residual for Dependent Variable Y2 (Scope)

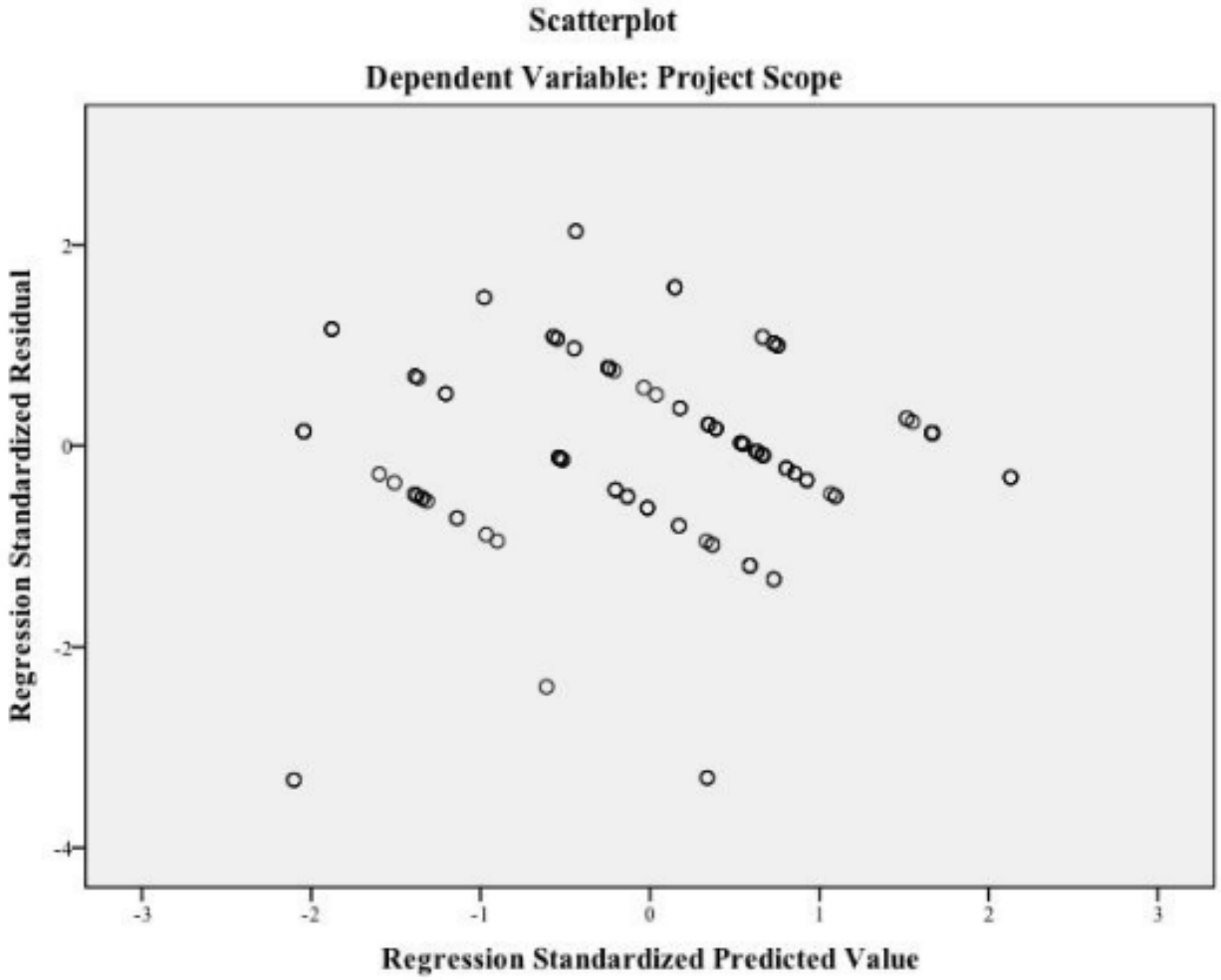

Figure 8. Normal Scatterplot of Regression Standardized Residual for Dependent Variable Y2 (Scope)

Table 15 shows the R and R^2 values for the full multiple regression analysis of the Y2 dependent variable. The multiple correlation coefficient R represents the linear correlation between the different variables (Laerd Statistics, 2015). The R-value of .707 suggests a strong relationship. R^2 represents the amount of change attributed to the independent variables. The R^2 shows that the model explained 50% of the total variance.

Table 17 shows the details of the coefficients for all 12 independent variables. The table shows both the unstandardized B coefficient and standardized B coefficients. The

unstandardized coefficients explain the contribution the variable made towards the prediction of Y2 when all other variables were held constant. The standardized coefficients describe the extent of each predictor variable's contribution to the joint prediction of Y2. The table also shows the probability of sample occurrence in testing the null hypothesis as reflected by the t-test.

Table 17

Coefficients of Scrum CSFs and Scope

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients		
		B	Std. Error	Beta	t	Sig.
1	(Constant)	1.687	.628		2.685	.008
	Management Commitment	-.015	.057	-.024	-.264	.792
	Organization Environment	-.176	.108	-.193	-1.639	.104
	Team Environment	.059	.097	.078	.608	.544
	Team Capability	.388	.144	.353	2.700	.008
	Customer involvement	.007	.079	.009	.090	.928
	Project Management Process	-.175	.135	-.186	-1.296	.198
	Project Definition Process	.078	.066	.095	1.184	.239
	Agile Software Engineering Techniques	.088	.121	.096	.728	.468
	Delivery Strategy	.449	.094	.453	4.777	.000
	Project Nature	.190	.099	.207	1.916	.058
	Project Type	-.197	.068	-.231	-2.882	.005
	Project Schedule	-.015	.073	-.016	-.206	.837

a. Dependent Variable: Project Scope

Four variables showed relative higher B and β values, a higher absolute t-value, along with a significance value at below .1, including:

- Team Capability (B = .388, β = .353, $t = 2.7$),
- Delivery Strategy (B= .449, β = .453, $t = 4.777$),

- Project Nature (B= .190, β = .207, *t* = 1.916), and
- Project Type (B= -0.197, β = -0.231, t = -2.882).

A significance value at below .1 supported rejecting the null hypothesis. However, the Project Type variable showed negative B and β values and a negative relationship with the Y2 dependent variable. This result suggests that the less defined the scope and requirements (Project Type), the lower the chances for project success in regard to the Scope dimension. The analysis of these results suggests three out of the 12 CSFs for dependent variable Y2, Team Capability, Delivery Strategy, and Project Nature were significant enough to be considered CSFs that contribute to the Scope dimension of project success. In summary, the relationship between the factors and dependent variable are as follows:

Project Success (Scope) = *f* (Team Capability, Delivery Strategy, Project Nature).

$R^2 = .50$

Team Capability: $\beta = .353$, $t = 2.7$

Delivery Strategy: $\beta = .453$, $t = 4.777$

Project Nature: $\beta = .207$, $t = 1.916$

Hypothesis Three Testing Results

$H3_0$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) do not relate to the time of agile software development projects.

$H3_a$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management

Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) significantly relate to the time of agile software development projects.

The following tables and figures summarize the results of data analysis for hypothesis three. Hypothesis was tested using linear regression and found that the Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, Project Schedule) were significantly related to the time of agile software development projects (Table 18 – ANOVA; alpha = 0.05; sig. < 0.000). Thus, the null hypothesis ($H3_0$) is rejected, and the alternate hypothesis ($H3_a$) accepted. An effect size analysis found a weak to medium relationship between Scrum CSFs and Quality (Gloeckner et al., 2001, p. 227; Table 19 – Model Summary; Adjusted R Square = 0.222). A visual inspection of residuals (Figures 9, 10, and 11) in histogram, P-P plot and scatter plot found there are no violations of the normality or linearity assumptions.

Table 18

ANOVA of Scrum CSFs and Time

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	77.637	12	6.470	4.117	.000[b]
	Residual	186.992	119	1.571		
	Total	264.629	131			

a. Dependent Variable: Project Time

b. Predictors: (Constant), Project Schedule, Delivery Strategy, Project Definition Process, Project Type, Management Commitment, Customer involvement, Project Nature, Organization Environment, Agile Software Engineering Techniques, Team Capability, Team Environment, Project Management Process

Table 19

Model Summary of Scrum CSFs and Time

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.542[a]	.293	.222	1.25354

a. Predictors: (Constant), Project Schedule, Delivery Strategy, Project Definition Process, Project Type, Management Commitment, Customer involvement, Project Nature, Organization Environment, Agile Software Engineering Techniques, Team Capability, Team Environment, Project Management Process

b. Dependent Variable: Project Time

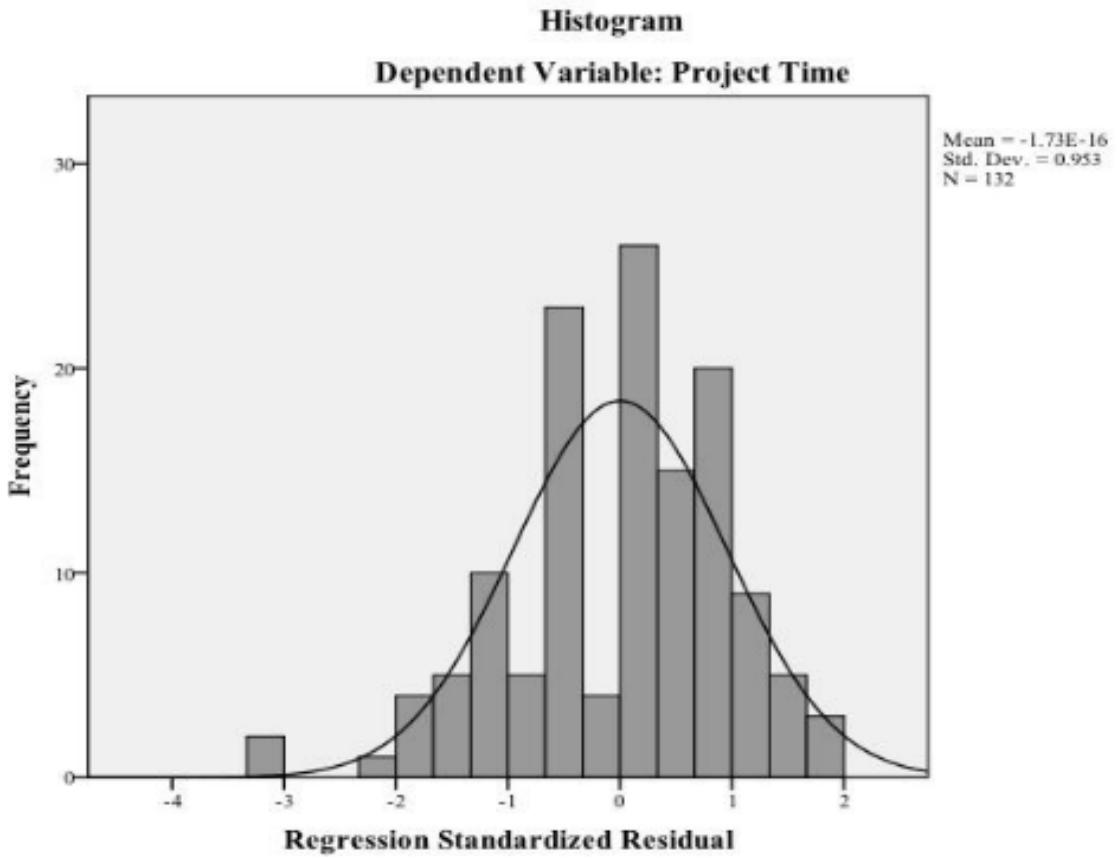

Figure 9. Histogram of the Regression Standardized Residual Model for Y3 (Time)

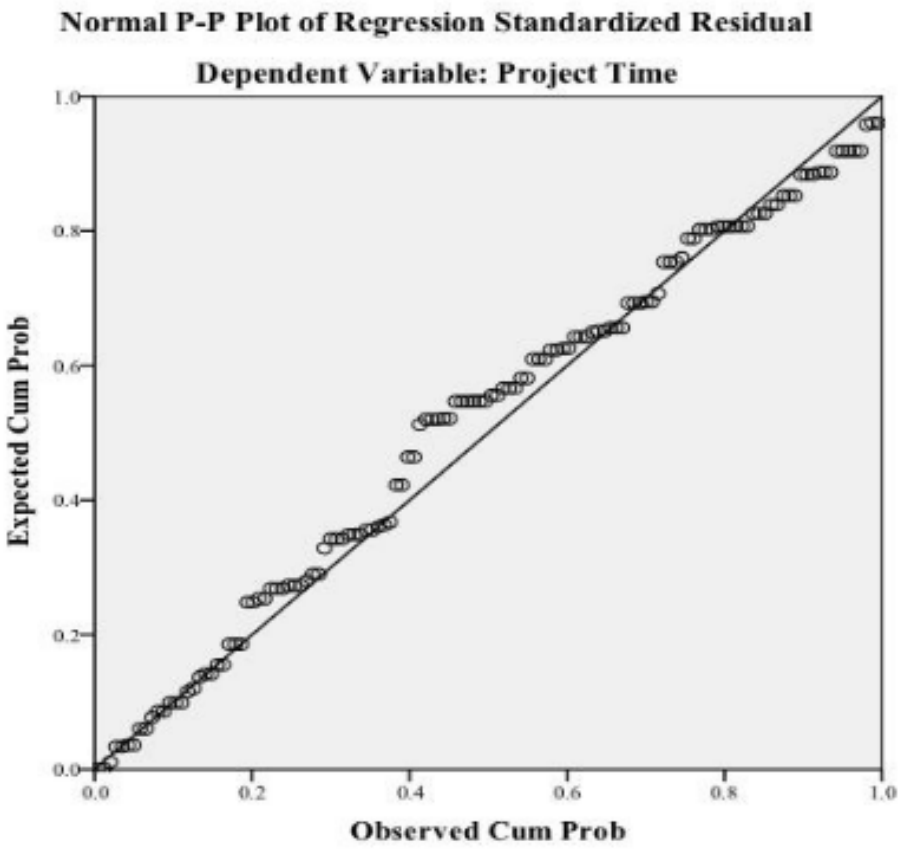

Figure 10. Normal P-P Plot of Regression Standardized Residual for Dependent Variable Y3 (Time)

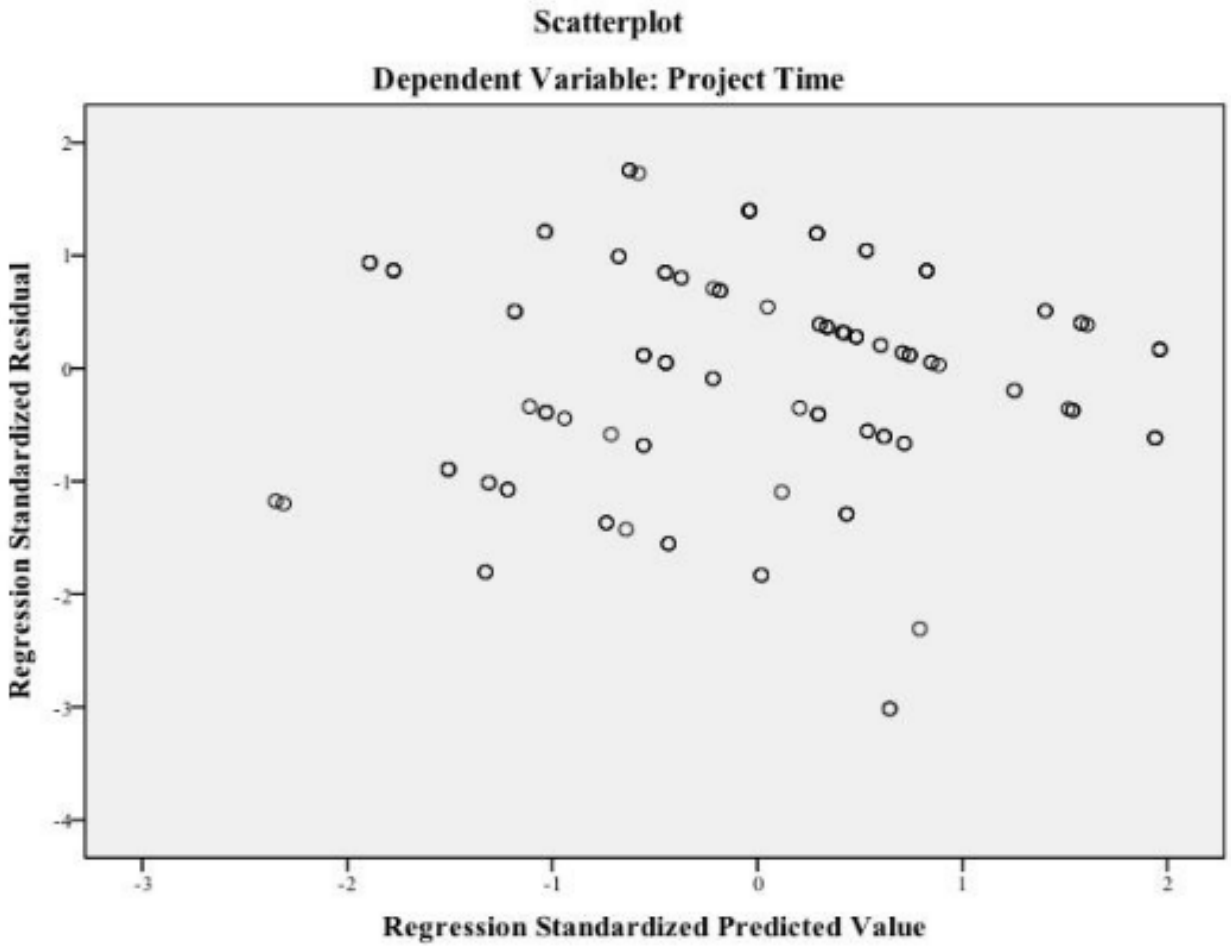

Figure 11. Normal Scatterplot of Regression Standardized Residual for Dependent Variable Y3 (Time)

Figure 9 shows the histogram of the residual in the regression model for the Y3 dependent variable, which suggests a normal distribution. Figure 10 shows the P-P plot for the residual of the regression model for the Y3 dependent variable, which is approximately a straight line. Figure 11 reflects the normal scatterplot for variable Y3. These tests suggest that the distribution of the Y3 dependent variable is almost normal and there are no violations of the normality or linearity assumptions.

Table 18 shows the R and R^2 values for the full multiple regression analysis of the Y3 dependent variable. The multiple correlation coefficient R represents the linear relationship

between the different variables (Laerd Statistics, 2015). The R-value of .542 suggests a strong correlation. R^2 represents the amount of attributed change to the independent variables. The R^2 shows that the model explained 29.3% of the total variance.

Table 20 shows the detailed information for all 12 independent variables. The table shows both the unstandardized B coefficient and standardized β coefficients. The first coefficient mentioned explains the difference in Y for each unit change in the predictor variable. The second coefficient explains the amount each predictor variable contributed to the overall prediction of Y3. The table also shows the probability of sample occurrence in testing the null hypothesis as reflected by the *t*-test. Four variables showed relative higher B and β values, a higher absolute t-value, along with a significance value at below .1, including:

- Management Commitment (B= -.152, β = .195, t = -1.804),
- Project Definition Process (B= .187, β = .185, t = 1.934),
- Delivery Strategy (B= .471, β = .384, t = 3.405), and
- Project Type (B= -.177, β = -.169, t = -1.768).

Table 20

Coefficients of Scrum CSFs and Time

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients		
		B	Std. Error	Beta	t	Sig.
1	(Constant)	3.309	.924		3.580	.000
	Management Commitment	-.152	.084	-.195	-1.804	.074
	Organization Environment	-.086	.158	-.076	-.545	.587
	Team Environment	.033	.142	.036	.232	.817
	Team Capability	.075	.212	.055	.353	.725
	Customer involvement	-.050	.116	-.049	-.433	.665
	Project Management Process	-.020	.198	-.018	-.103	.918
	Project Definition Process	.187	.097	.185	1.934	.056
	Agile Software Engineering Techniques	.154	.178	.135	.863	.390
	Delivery Strategy	.471	.138	.384	3.405	.001
	Project Nature	.080	.146	.071	.550	.583
	Project Type	-.177	.100	-.169	-1.768	.080
	Project Schedule	-.127	.107	-.108	-1.187	.238

a. Dependent Variable: Project Time

A significance value at below .1 supported rejecting the null hypothesis. However, the Management Commitment and Project Type variables showed negative B and β values and a negative relationship with the Y3 dependent variable. These results suggest that less management involvement (Management Commitment) and less defined scope and flexibility (Project Type) contribute to the increased Time dimension resulting in lower chances of project success.

Regarding the variable Y3, only two factors – Project Definition Process, and Delivery Strategy, were significant enough to be considered CSFs that contribute to the Time dimension

of project success. In summary, the relationship between the factors and dependent variable are as follows:

Project Success (Time) = f (Project Definition Process, Delivery Strategy).

$R^2 = .293$

Project Definition Process: $\beta = .185$, $t = 1.934$

Delivery Strategy: $\beta = .384$, $t = 3.405$

Hypothesis Four Testing Results

$H4_0$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) do not relate to the cost of agile software development projects.

$H4_a$: Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) significantly relate to the cost of agile software development projects.

The following tables and figures summarize the results of data analysis for hypothesis four. Hypothesis was tested using linear regression and found that the Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, Project Schedule) were significantly related to the cost of agile software development projects (Table 22 –

ANOVA; alpha = 0.05; sig. < 0.000). Thus, the null hypothesis ($H4_0$) is rejected, and the alternate hypothesis ($H4_a$) accepted. An effect size analysis found a medium relationship between Scrum CSFs and Quality (Gloeckner et al., 2001, p. 227; Table 21 – Model Summary; Adjusted R Square = 0.355). A visual inspection of residuals (Figures 12, 13, and 14) in histogram, P-P plot and scatter plot found there are no violations of the normality or linearity assumptions.

Table 21

Model Summary of Scrum CSFs and Cost

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.644[a]	.414	.355	1.22831

a. Predictors: (Constant), Project Schedule, Delivery Strategy, Project Definition Process, Project Type, Management Commitment, Customer involvement, Project Nature, Organization Environment, Agile Software Engineering Techniques, Team Capability, Team Environment, Project Management Process
b. Dependent Variable: Project Cost

Table 22

ANOVA of Scrum CSFs and Cost

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	127.087	12	10.591	7.019	.000[b]
	Residual	179.542	119	1.509		
	Total	306.629	131			

a. Dependent Variable: Project Cost

b. Predictors: (Constant), Project Schedule, Delivery Strategy, Project Definition Process, Project Type, Management Commitment, Customer involvement, Project Nature, Organization Environment, Agile Software Engineering Techniques, Team Capability, Team Environment, Project Management Process

Figure 12 shows the histogram of the residual in the regression model for the Y4 dependent variable, which suggests a normal distribution. Figure 13 shows the P-P plot for the residual of the regression model for the Y4 dependent variable, which is approximately a straight line. Figure 14 represents the scatterplot distribution for variable Y4. These tests suggest that the distribution of the Y4 dependent variable is almost normal and there are no violations of the normality or linearity assumptions.

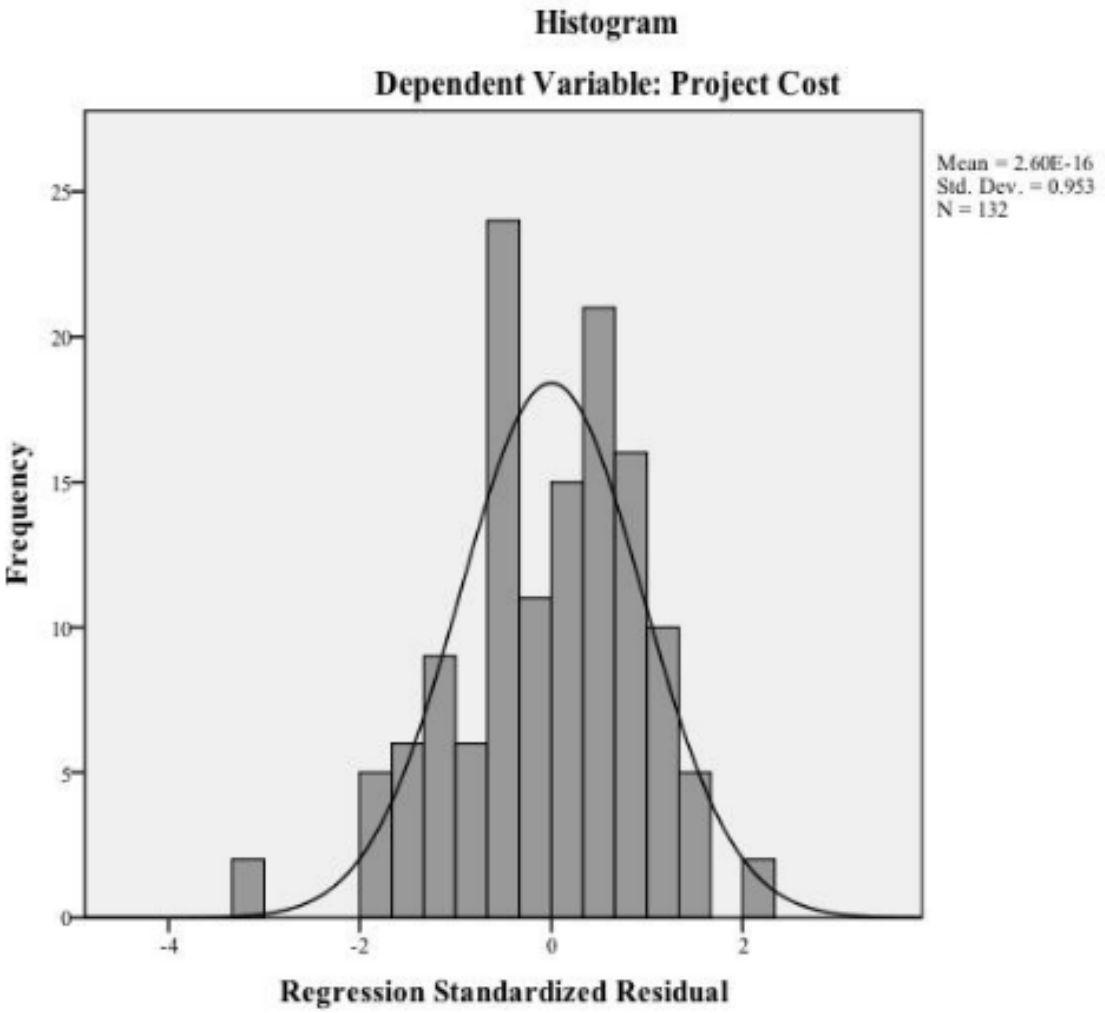

Figure 12. Histogram of the Regression Standardized Residual Model for Y4 (Cost)

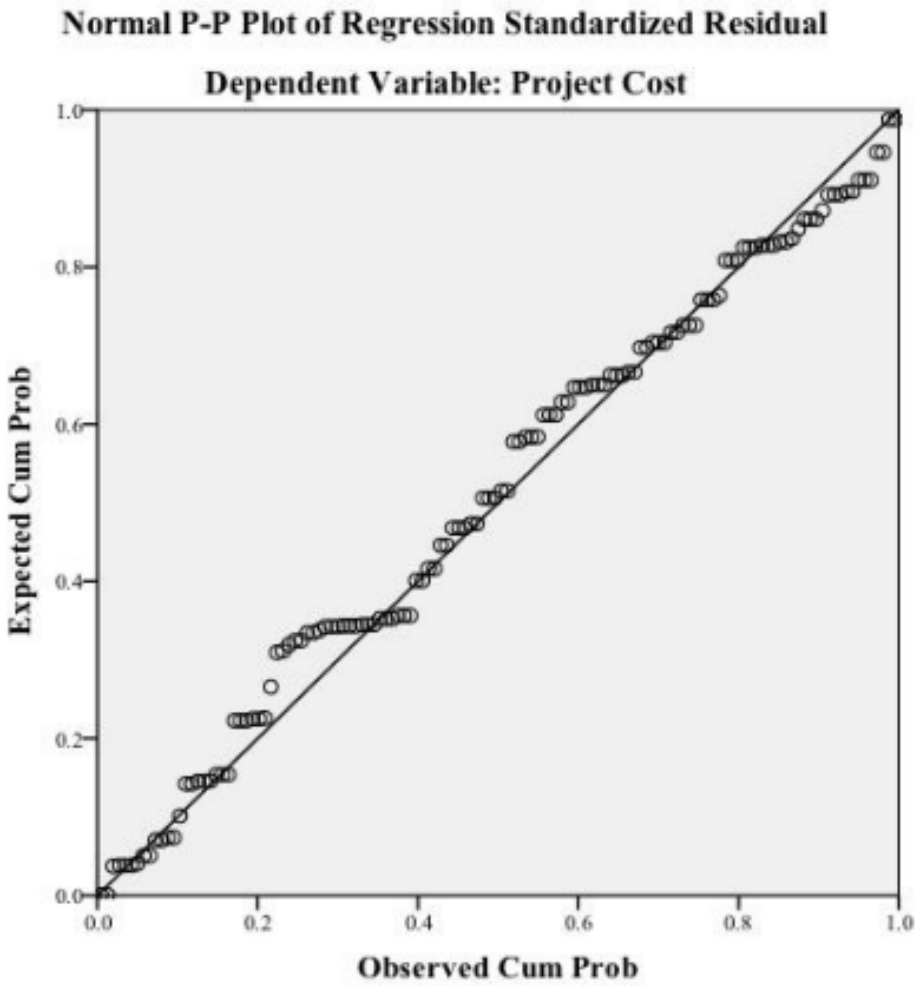

Figure 13. Normal P-P Plot of Regression Standardized Residual for Dependent Variable Y4 (Cost)

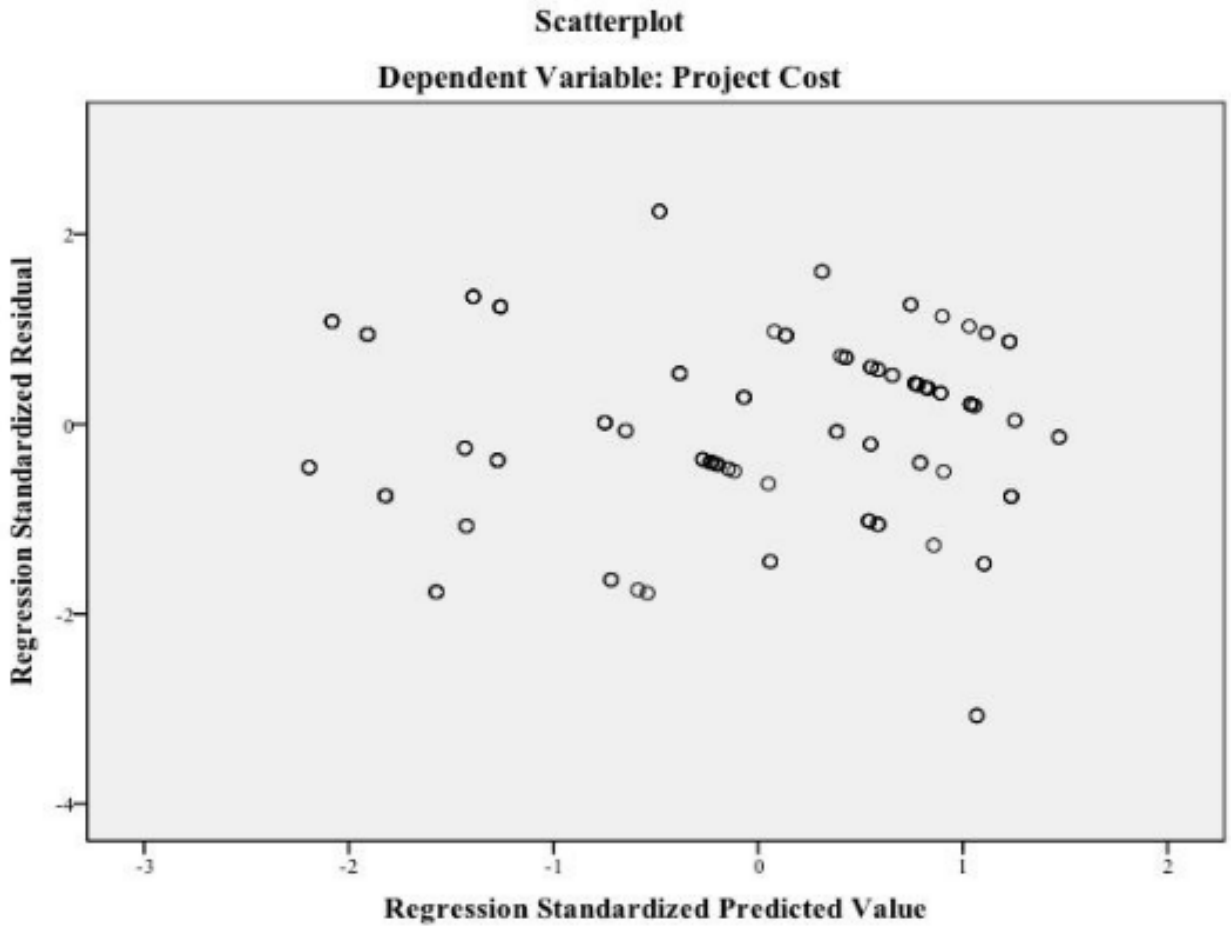

Figure 14. Normal P-P Plot of Regression Standardized Residual for Dependent Variable Y4 (Cost)

Table 21 shows the R and R^2 values for the full multiple regression analysis of the Y4 dependent variable. The multiple correlation coefficient R represents the linear correlation between the different variables (Laerd Statistics, 2015). The R-value of .644 suggests a strong relationship. R^2 represents the amount of change attributed to the independent variables. The R^2 shows that the model explained 35.5% of the total variance.

Table 23 shows the detailed information for all 12 independent variables. The table shows both the unstandardized B coefficient and standardized β coefficients. The first coefficient mentioned explains the difference in Y for each unit change in in the predictor

variable. The second coefficient explains the amount each predictor variable contributed to the overall prediction of Y4. The table also shows the probability of sample occurrence in testing the null hypothesis as reflected by the *t*-test. Four variables showed relative higher B and β values, a higher absolute t-value, along with a significance value at below .1, including:

- Customer Involvement (B= -.426, β = -.39, t = -3.749),
- Project Management Process (B= 0.417, β = 0.333, *t* = 2.145),
- Project Definition Process (B= 0.344, β = 0.317, *t* = 3.633), and
- Project Nature (B= -.336, β = -.275, t = -2.355).

A significance value at below .1 supported rejecting the null hypothesis. However, the Customer Involvement and Project Nature variables showed negative B and β values and a negative relationship with the Y4 dependent variable. These results suggest that less involvement by the customer (Customer Involvement) and/or building non-life-critical software product (i.e., not advanced weapons programs or air traffic control programs) (Project Nature) contribute to increasing Cost dimension and reducing the chances of project success. Non-life-critical could be business critical software.

Table 23

Coefficients of Scrum CSFs and Cost

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients		
		B	Std. Error	Beta	t	Sig.
1	(Constant)	2.170	.906		2.395	.018
	Management Commitment	.054	.082	.065	.658	.512
	Organization Environment	.158	.155	.129	1.016	.312
	Team Environment	.010	.139	.010	.069	.945
	Team Capability	.113	.207	.077	.544	.587
	Customer involvement	-.426	.114	-.390	-3.749	.000
	Project Management Process	.417	.194	.333	2.145	.034
	Project Definition Process	.344	.095	.317	3.633	.000
	Agile Software Engineering Techniques	.166	.175	.135	.949	.344
	Delivery Strategy	.108	.135	.082	.797	.427
	Project Nature	-.336	.143	-.275	-2.355	.020
	Project Type	.089	.098	.079	.907	.366
	Project Schedule	-.107	.105	-.084	-1.018	.311

a. Dependent Variable: Project Cost

Regarding the variable Y4, only the two factors – Project Management Process and Project Definition Process, were significant enough to be considered CSFs that contribute to the Cost dimension of project success. In summarization, the relationship between the factors and the dependent variable is as follows:

Project Success (Cost) = f (Project Management Process, Project Definition Process).

$R^2 = .414$

Project Management Process: $\beta = .333$, $t = 2.145$

Project Definition Process: $\beta = .317$, $t = 3.633$

Summary

The final results are described in the analysis using the full regression model previously discussed. The full regression model consisted of all 12 independent variables entered for each dependent variable. Table 24 shows the final list of the most significant independent variables (CSFs) for the dimensions of project success (Quality, Scope, Time, and Cost).

Table 24

Top CSFs by frequency and value

Rank	Factor	Frequency	*B* value
1	Delivery Strategy	3	0.254, 0.453, 0.384
2	Team Capability	2	0.460, 0.353
3	Project Definition Process	2	0.185, 0.317
4	Project Management Process	1	0.333
5	Project Nature	1	0.207

Based on these findings, Delivery Strategy is the most significant CSF. Then the next most significant CSFs are Team Capability and Project Definition Process. Team Capability has higher β values than Project Definition Process. Also, Project Management Process and Project Nature came in next with Project Management Process ranking a little higher. The remaining seven variables (CSFs) did not show a significant value in the regression model, therefore not considered CSFs. The full regression model consisted of all 12 independent variables entered for each dependent variable. Rejection of all null hypotheses, and acceptance of the alternatives occurred for each of the four questions. The relationships in the model range from weak to strong for each of the four dimensions of project success – Quality, Scope, Time and Cost.

CHAPTER 5. CONCLUSIONS

Introduction

Given that Scrum has become the most popular agile software development methodology, scholars and practitioners (Dyba & Dingsoyr, 2008; Ghani et al., 2015; Papatheocharous &Andreou, 2014; VersionOne, 2016a) have recommended further research on its implementation in agile software development projects. Scholars have also suggested research focusing on examining CSFs for the implementation of Scrum methodology in global companies (Brown, 2015) and within distributed teams (Matalonga et al., 2013), which are two areas of focus in this study. Chow and Cao (2008), who pioneered research on CFSs for agile software development projects, recommended expanding their research model with a focus on specific agile software development methodologies and after a longer period of maturation and exposure to these methodologies.

Expanding on Chao and Cao and Brown's studies, the purpose of this study was to examine the relationships between 12 independent variables (representing possible CSFs for agile software development projects) and the dependent variable of project success (consisting of four dimensions) in large and distributed software development projects using Scrum methodology in U.S.-based global companies. This study contributes to the existing body of knowledge in the field of project management given that examining CSFs for agile software development projects continues to be a relevant topic of inquiry. This chapter serves as an evaluation of the findings of the study in consideration of its research questions, fulfillment of its

research purpose, and contributions to the business problem. The chapter also addresses the study's conclusions and recommendations for future research.

Evaluation of Research Questions

In answering the research questions, the full regression model served to analyze data collected from study participants. Four dimensions – Quality, Scope, Time, and Cost, served as criteria for measuring participants' perceptions regarding project success. Based on the outcome of the analysis of statistical results and hypotheses described in Chapter 4, this section serves to provide answers to the study's research questions.

Research Question 1

RQ1: To what extent do Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) predict the quality of agile software development projects?

Results reflected in the multiple regression analysis conducted in the study support that the 12 CSFs are significantly related to the quality of agile software development projects using Scrum methodology in U.S.-based global companies. However, results for two of the CSFs – Team Capability and Delivery Strategy, showed stronger relationships to the Quality dimension of project success than other factors. Team Capability suggests the importance of a Scrum team having a high level of competence, expertise, and motivation; and a software development project having a well-defined scope. Delivery Strategy entails both delivering software at a regular pace and delivering the most important features first, which suggest that this is an

important aspect of the success of agile software development projects using Scrum methodology.

Research Question 2

RQ2: To what extent do Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) predict the scope of agile software development projects?

Results reflected in the multiple regression analysis conducted in the study support that the 12 CSFs are significantly related to the scope of agile software development projects using Scrum methodology in U.S.-based global companies. Three of the 12 CSFs – Team Capability, Delivery Strategy, and Project Nature, showed strong and significant relationships with the Scope dimension of project success. As a top CSF, Delivery Strategy, had a high β value for the full regression model. Delivery Strategy entails both delivering software at a regular pace and delivering the most important features first. Next, Team Capability and Project Nature, are both important. Team Capability entails the importance of a Scrum team having a high level of competence, expertise, and motivation. Project Nature highlights principles of agile methodologies such as being people-centric, promoting self-organizing teamwork, delivering product features continuously, and requiring minimal documentation.

Research Question 3

RQ3: To what extent do Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software

Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) predict the time of agile software development projects?

Results reflected in the multiple regression analysis conducted in the study support that the 12 CSFs are significantly related to the time of agile software development projects using Scrum methodology in U.S.-based global companies. Two of the 12 CSFs – Project Definition Process and Delivery Strategy, showed strong and significant relationships with the Time dimension of project success. Project Definition Process entails a software development project having a well-defined scope and featuring up-front risk analysis and cost review. Delivery Strategy entails both delivering software at a regular pace and delivering the most important features first.

Research Question 4

RQ4: To what extent do Scrum CSFs (Management Commitment, Organization Environment, Team Environment, Team Capability, Customer Involvement, Project Management Process, Project Definition Process, Agile Software Engineering Techniques, Delivery Strategy, Project Nature, Project Type, and Project Schedule) predict the cost of agile software development projects?

Results reflected in the multiple regression analysis conducted in the study support that the 12 CSFs are significantly related to the cost of agile software development projects using Scrum methodology in U.S.-based global companies. Two CSFs – Project Management Process and Project Definition Process, showed stronger relationships to the Cost dimension of project success than other factors. Project Management Process entails having good processes for requirement management and tracking as well as ongoing communication. Project Definition

Process refers to a software development project that has a well-defined scope and features up-front risk analysis and cost review.

Another view presented in Table 25 is a review and evaluation of the most significant CSFs for each of four dimensions of project success (Quality, Scope, Time, and Cost). First, for the Quality dimension of project success, there were two significant CSFs supported by the full regression model, including Team Capability and Delivery Strategy. These two CSFs contribute to the quality of deliverables and overall project success. Quality improves with regular delivery of the most important items by highly qualified and competent teams that are motivated and knowledgeable.

Second, for the Scope dimension of project success, there were three significant CSFs supported by the full regression model, including Team Capability, Delivery Strategy, and Project Nature. These CSFs are interrelated. Delivery Strategy supports regular delivery of the highest priority requirements. Team Capability supports a highly competent team that is motivated. Project Nature supports the principles of agile methodologies of being people-centric, promoting self-organizing teamwork, delivering product features continuously, and requiring minimal documentation. These three CSFs support the Scope dimension of project success by contributing to the delivery of products quickly, and in a timely and consistent basis.

Third, for the Time dimension of project success, there were two significant CSFs supported by the full regression model, including Project Definition Process and Delivery Strategy. Project Definition Process supports a well-defined scope with up-front risk and cost analysis. Delivery Strategy supports regular delivery of the highest priority requirements. These two CSFs support reducing the time of the project when addressed in advance and consistently throughout the agile software development process.

Table 25

Results of 12 CSFs by Dimension

Factor	Dimension			
	Y1 - Quality	Y2 - Scope	Y3 - Time	Y4 - Cost
CSF Category - Organizational Factors/ Dimension				
CSF1 - Management Commitment				
CSF2 - Organizational Environment				
CSF3 - Team Environment				
CSF Category - People Factors/ Dimension				
CSF4 - Team Capability	X	X		
CSF5 - Customer Involvement				
CSF Category - Process Factors/ Dimension				
CSF6 - Project Management Process				X
CSF7 - Project Definition Process			X	X
CSF Category - Technical Factors/ Dimension				
CSF8 - Agile Software Techniques				
CSF9 - Delivery Strategy	X	X	X	
CSF Category - Project Factors/ Dimension				
CSF10 - Project Nature		X		
CSF11 - Project Type				
CSF12 - Project Schedule				

Note: "X" represents the CSF is significant to the corresponding dependent variable.

Finally, for the Cost dimension of project success, there were two significant CSFs supported by the full regression model, including Project Management Process and Project Definition Process. These two CSFs contribute to reducing the overall cost of the project and overall project success. Both Project Management Process and Project Definition Process suggest the need for identifying the scope of the project and requiring minimal documentation.

Based on the analysis of regression results and hypotheses tests described in Chapter 4, the differences were evident among the five categories of CSFs (Organizational, People, Process, Technical, and Project) regarding their predictive power for various dimensions of project success. First, the Technical category was the most critical for predicting project success, by covering three of the four dimensions (Quality, Scope, and Time). Two other categories, including People and Process, each touched two of the project success dimensions. The Process category covered the project success dimensions of Time and Cost. The People category covered the project success dimensions of Quality and Scope. The Project category covered the project success dimension of Scope. Finally, the Organizational category did not cover any dimension of project success.

In summary, the Technical category is the most important for supporting ongoing delivery of valuable product and project success regarding the Quality, Scope, and Time dimensions of project success. Next, the People and Process categories support defining the right scope, creating the right amount of documentation, and creating a self-organizing team of competent and motivated professionals. Lastly, the Project category proposes creating organizational and team climate for improving project success regarding the Scope dimension.

Fulfillment of Research Purpose

This was an explanatory, quantitative, and survey study for examining the research model proposed by Chow and Cao (2008), which hypothesizes about the relationships between 12 independent variables, representing possible CSFs for agile software development projects, and the dependent variable of project success consisting of four dimensions. The focus of the study was on examining these relationships within the context of large and distributed agile software

development projects using Scrum methodology in U.S.-based global companies. Findings of the study, including regression analysis, hypothesis testing, and the evaluation of the research questions, provide evidence that fulfillment of the research purpose occurred.

As reflected in Chapter 4, the study's findings support that all of the 12 CSFs initially identified by Chow & Cao (2008) have an impact on the successful resolution of agile software development projects using Scrum methodology in U.S.-based global companies; however, with differing levels of significance. The results reflect all 12 factors (independent variables) are not significant CSFs for one or more of four dimensions of project success (dependent variable). Also, five of the 12 CSFs are significant; however, of these CSFs, three ranked higher than the others and had significant impact on more than one of the dimensions of project success. These three factors are Delivery Strategy, Team Capability, and Project Definition Process.

Of the five categories of CSFs, three of them, Technical, Process, and People showed the greatest impact on overall success. Among these, Delivery Strategy is particularly important for the Scrum process to be successful, which was also apparent in the findings from Chow and Cao (2008) and Brown's (2015) studies. Delivery Strategy is a CSF for agile software development projects using Scrum methodology that accounts for delivery of the most important features first. Moreover, developing competent and self-motivated teams, delivering the highest priority features in a consistent and timely manner, and requiring minimal documentation contribute to overall project success. These areas must be done correctly for achieving success in software developing projects using Scrum methodology.

Given the fact that this study is an expansion of an existing research model, it is important to compare its findings from those of the previous studies. First, in the research conducted by Chow and Cao (2008), the participants were mainly located outside the U.S. and

used XP as the chosen methodology; whereas participants in this study were located around the world including the U.S. but used only the Scrum methodology. The results from Chow and Cao's study supported 10 of the 48 hypotheses, with only six factors found to be significant and considered CSFs for agile software development projects. These six factors were Agile Software Engineering Techniques, Team Environment, Project Management Process, Delivery Strategy, Customer Involvement, and Team Capability.

When comparing the results of previous research with the current study, some of the differences may be because of the different methods represented, Scrum vs. XP, in each. Scrum and XP are similar in approach as both features the delivery of product in set interactions. Another aspect of consideration is the maturity of Scrum, which has been used by practitioners for over 15 years, and now becoming the most commonly used agile software development methodology. The seasoning of the practitioners could account for the two CSFs not previously identified – Project Definition Process and Project Nature. These CSFs stress the importance of identifying the best methodology for the organization, being people-centric, promoting self-organizing teamwork, and requiring minimal documentation.

Second, the research conducted by Brown (2015) concentrated on participants that resided only in the U.S., most of whom were users of the XP methodology; whereas this study engaged participants in U.S.-based global companies and focused only on users of the Scrum methodology. Results of the survey conducted by Brown – like Chow and Cao (2008), found that only six of the 12 factors were significant and considered CSFs for agile software development projects. These six factors included Project Type, Project Schedule, Project Nature, Management Commitment, Project Definition Process, and Delivery Strategy.

Between the six CSFs that Chow and Cao (2008) found significant and the six that Brown (2015) found significant, only one was the same – Delivery Strategy. This study found that five factors were significant and considered CSFs for agile software development projects using Scrum methodology in U.S.-based global companies. Similar to the findings of Chow and Cao and Brown, this study supports that Delivery Strategy is the most significant CSF. This information stresses the importance of securing continuous delivery of the most important software features.

Third, Stankovic et al. (2013), who also replicated Chow and Cao's (2008) research model, used a smaller sample (23 participants) representing senior developers and project managers in IT companies in Yugoslavia. In contrast, this study had a larger sample (132 participants) representing project managers and agile software development practitioners that are users of Scrum methodology and working for U.S.-based global companies. The study conducted by Stankovic et al. reflected only a few of the CSFs as significant. These include Project Management Process, Project Definition Process, Project Nature, and Project Schedule. Whereas, the current study supports five of the 12 factors as significant. These include three of the CSFs identified by Stankovic et al. – Project Management Process, Project Definition Process, and Project Nature. This support that project success is dependent on using the right methodology, developing organizational/team climate, being people-centric, promoting self-organizing teamwork, prioritizing product features, and requiring minimal documentation.

Fourth, given the challenges encountered by global organizations when utilizing Scrum methodology as discussed in Chapter 2, this study contributes to identifying the areas that are critical to project success. Findings of the study are significant to practitioners in the field of project management, particularly those using agile methods and Scrum methodology for large

and distributed software development projects in U.S.-based global companies. Considering CSFs serve to focus efforts in a few areas that when achieved can lead to overall project success (Bullen & Rockart, 1981), this study helps to direct attention to a few specific areas. It appears that whether the project is large and distributed or smaller, many of the same CSFs are important for overall project success.

Now that the use of agile software development methodologies has matured, the results of the study reflect that five of the 12 of the factors proposed by Chow and Cao (2008) are significant and critical for the success of agile software development projects, particularly those considered large and distributed and using Scrum methodology. However, Delivery Strategy has an impact on more of the dimensions of project success. Delivery Strategy includes regular delivery of features and delivering the most important feature first, which are main expectations of customers. The other factors support this end goal.

Finally, it is important to note that as with any research study, this one has certain limitations. These limitations include the fact that representation of an organization or industry did not exist in the study. Also, participants may have a different understanding and interpretation of what a large project is. When using surveys, there is always the possibility of participants' subjective biases on the determination of project success. Lastly, the instrument itself could have better-captured participant demographics, with a few changes to the questions along with having multiple choice answers.

Contribution to Business Problem

This study's general problem as identified in Chapter 1 is that as U.S.-based global companies increase and scale their use of agile software development methodologies, the rate of

success of projects has been lower than expected (Brown, 2015; Hastie & Wojewoda, 2015; Senapathi & Srinivasan, 2012). More specifically, managers are confronted with challenges when implementing Scrum methodology in large agile software development projects with distributed project teams (Gandomani et al., 2014; Gonçalves & Lopes, 2014). One of these challenges is that management and teams use the methodology without sufficient executive sponsorship and planning. Challenges also compound issues created by geographical distribution and cross-functionality of teams, along with the over-emphasis on quick results with minimal amounts of testing (Cao et al., 2009; Fernandez & Fernandez, 2008; Hoda & Murugesan, 2016; Khalil & Khalil, 2016).

This study contributes to the body of knowledge in the field of project management by pointing out significant CSFs that practitioners can apply to avoid and/or mitigate those challenges commonly faced by organizations when implementing or scaling the use of agile methods, particularly the Scrum methodology, for their software development projects. Findings of the study support the foundation established with the creation of the *Agile Manifesto*. Of the six CSFs supported by the findings of this study, the most important is Delivery Strategy. The supported correlation with the *Agile Manifesto* is as follows. Delivery Strategy relates to satisfying the customer with continuous delivery and delivering software frequently. The support that the study gives to the value of the *Agile Manifesto* reiterates the importance of its core values.

Findings of the study contribute to the business problem resolution and guidance. By guiding companies in the areas that should have the most focus throughout the agile software development process, reduction of challenges should happen while providing overall project success. The results of the study support that five of the 12 factors proposed by Chow and Cao

(2008) are CSFs for large and distributed agile software development projects using Scrum methodology in U.S.-based global companies. Nevertheless, some of the five CSFs have stronger relationships to project success, including Delivery Strategy, Team Capability, and Project Definition Process. The Technical category of CSFs is the highest ranking regarding project success dimensions.

Table 25 lists the four different dimensions of project success (Quality, Scope, Time, and Cost) with the CSFs identified in the results. The Quality dimension of project success is supported by the CSFs of Team Capability and Delivery Strategy, which propose accepting the agile methodology, creating high-caliber teams, delivering of product features regularly, and being flexible in managing scope. The Scope dimension of project success is supported by these two CSFs as well as Project Nature. The additional CSF, Project Nature, supports using agile processes for delivering the most important scope. The Time dimension of project success is supported by the CSFs of Project Definition Process and Delivery Strategy. These factors propose establishing a clear scope, following a process, and being flexible. Lastly, the Cost dimension of project success is supported by Project Management Process and Project Definition Process. Therefore, if the team is strong, high-caliber, and focused on working with the customer and following processes, a quality product can be delivered quickly to meet the needs of the customer; hence keeping costs low.

Agile methodologies serve to improve the speed and quality of the delivered product with small teams as highlighted in the *Agile Manifesto*. Converting from traditional project management to agile methods is beneficial to organizations in the current business environment requiring quick response to change. However, completion of the foundational processes and due-diligence must still occur before beginning to ensure a smooth process. Management

commitment is not a CSF; however, as long as there is cooperation of the teams and acceptance of the agile process, the lack of executive support has minimal impact.

This study contributes to project management's body of knowledge by identifying those areas that are necessary for overall project success to occur. By focusing on CSFs, management and practitioners of the Scrum methodology will be able to focus improvement efforts towards the areas that are lacking in the process while expanding overall knowledge of the Scrum team.

Recommendations for Further Research

There are four areas for future research supported by the evaluation of findings of this study. First, findings of the study, with 39.4% of the participants using SAFe, suggest that the SAFe process is becoming more popular as a programming tool to use with the Scrum methodology. A large number (39%) of participants in this study reported that they had used a combination of SAFe and the Scrum methodology for their agile software development projects. Utilization of SAFe is relatively new and should be an avenue of future research to determine if its use contributes to the success of large software development projects. SAFe is a new framework added on top of an agile methodology and should be explored to determine its value to the overall project success within an organization.

Second, this was a quantitative study focused on global organizations; therefore, it would be recommended to expand the research with a qualitative study or possibly a case study of one organization to substantiate the results found in previous studies. These other research approaches may bring more insight by describing how practitioners have applied CSF theory to drive success in agile software development projects. For instance, a case study for one company would give more detail to that organization and possibly the industry regarding the

application of CSFs. A qualitative study with business leaders may also expound on the perspective of project success and the achievement within different organizations. A qualitative study would provide more detail on the different CSFs to help determine the real value of each factor.

Third, one avenue for future research to address limitations of this study is applying Chow and Cao's (2008) research model to examine CSFs for agile software development projects using other popular agile software development methodologies such as Kanban. Future studies for expanding on Chow and Cao's research model may benefit from modifications to the survey instrument, including better categorization of choices for the different areas in demographics and project details.

Lastly, a comparison of different user groups could build on this study's results. For example, a future study could compare the views of management to those of their teams. Another approach would be to compare teams in different countries or industry sectors with each other to determine if the application of CSFs for agile software development projects differ and/or produces varying results.

Conclusions

This explanatory, quantitative, and survey study focused on examining the significance of 12 CSFs for the implementation of Scrum methodology in large and distributed software development projects using Scrum methodology in U.S.-based global companies. Previous research (Brown, 2015; Chow & Cao, 2008) provided theoretical, topical, and methodological foundations, and the opportunity to expand and further test a research model for examining CSFs for agile software development projects in a different context. More specifically, the study

reflects the perspectives of project managers and software development practitioners regarding potential CSFs and success criteria in completed large and distributed agile software development projects using Scrum methodology in U.S.-based global companies. A survey instrument was used to collect data from 132 practitioners that had served as product owner, Scrum master, software developer, business analyst, and/or tester for a completed large and distributed agile software development project, using Scrum methodology, in U.S.-based global companies.

Study data analysis served to arrive to the following conclusions. Similar findings from previous research (Brown, 2015; Chow & Cao, 2008), this study found that five of the 12 potential CSFs are significant predictors of the success of large and distributed software development projects using Scrum methodology in U.S.-based global companies. The results revealed the correlation of each of the CSFs to each of the four different dimensions of project success it supported. The Quality dimension correlated with CSFs of Team Capability and Delivery Strategy. The Scope dimension correlated with the CSFs of Team Capability, Delivery Strategy, and Project Nature. The Time dimension correlated with Project Definition Process, and Delivery Strategy. Finally, the Cost dimension correlated with the CSFs of Project Management Process, and Project Definition Process.

These results provide guidance on what practitioners of the Scrum methodology should focus on to improve each dimension of project success. Even though only five of the 12 CSFs tested to be significant contributors to the success of agile software development projects, some of them have greater value and significance in contributing to project success. The three highest ranking factors are Delivery Strategy, Team Capability, and Project Definition Process. The

remaining two, including Project Management Process, and Project Nature, are important but may not prove to be as impactful to overall project success.

REFERENCES

Abrahamson, P., Warsta, J., Sippon, S. T., & Ronkainen, J. (2003). New directions on agile methods: A comparative analysis. *Proceedings of the 25th International Conference on Software Engineering, USA*, 244-254. Portland, OR. doi:10.1109/ICSE.2003.1201204

Agile Alliance (2016). *What is agile?* Retrieved from https://www.agilealliance.org/agile101/

Alexander, M. (2015, Oct). 5 practical project management certifications. *CIO from IDG*. Retrieved from http://www.cio.com/article/2992100/project-management/5-practical-project-management-certifications.html?page=5

Allen, I. E., & Seaman, C. A. (2007). Likert scales and data analyses. *Quality Progress, 40*(7), 64-65. Retrieved from https://www.scribd.com/document/94205656/Likert-Scales-and-Data-Analyses

Ambler, S. (2014a). *2014 Agile adoption survey results.* Retrieved from http://www.ambysoft.com/surveys/agileJanuary2014.html

Ambler, S. (2014b). *Feature driven development (FDD) and agile modeling*. Retrieved from http://www.agilemodeling.com/essays/fdd.htm

Anantatmula, V., & Thomas, M. (2010). Managing global projects: A structured approach for better performance. *Project Management Journal, 41*(2), 60-72. doi:10.1002/pmj.20168

Bannerman, P. L. (2008, July). *Defining project success: A multilevel framework*. Paper presented at PMI® Research Conference: Defining the Future of Project Management, Warsaw, Poland. Newtown Square, PA: Project Management Institute. (pp. 1-14). Retrieved from http://www.pmi.org/learning/library/defining-project-success-multilevel-framework-7096

Boynton, A. C., & Zmud, R. W. (1986). An assessment of critical success factors. *Sloan Management Review*, *25*(4), 17-27. Retrieved from https://www.researchgate.net/publication/282370599_An_Assessment_of_Critical_Success_Factors

Brown, G. A. (2015). *An examination of critical success factors of an agile project* (Doctoral dissertation). Retrieved from ProQuest Dissertations & Theses Global. (1660970158). (UMI No. 3684950)

Bullen, C. V., & Rockart, J. F. (1981). *A primer on critical success factors* (Working Paper No. 69). Cambridge, MA: Massachusetts Institute of Technology. Retrieved from http://18.7.29.232/bitstream/handle/1721.1/1988/SWP-1220-08368993-CISR-069.pdf?sequence=1

Burgan, S. C. & Burgan, D. S. (2014). *One size does not fit all: Choosing the right project approach.* Paper presented at PMI® Global Congress 2014—North America, Phoenix, AZ. Newtown Square, PA: Project Management Institute. Retrieved from https://www.pmi.org/learning/library/choosing-right-project-approach-9346

Campanelli, A. S., & Parreiras, F. S. (2015). Agile methods tailoring - A systematic literature review. *The Journal of Systems and Software, 110*, 85-100. doi:10.1016/j.jss.2015.08.035

Cao, D. (2006). An empirical investigation of critical success factors in agile software development projects (Doctoral dissertation). Retrieved from ProQuest Dissertations & Theses Global. (UMI No. 3238566)

Cao, L., Mohan, K., Xu, P., & Ramesh, B. (2009). A framework for adapting agile development methodologies. *European Journal of Information Systems, 18*(4), 332-343. doi:10.1057/ejis.2009.26

Chikhale, M., & Mansouri, M. (2015). An agile and collaborative framework for effective governance to enhance management in large-scale enterprise business systems: The case of Apple Inc. Global. *Journal of Flexible Systems Management, 16*(3), 283-293. doi:10.1007/s40171-015-0098-9

Chow, T., & Cao, D. (2008). A survey study of critical success factors in agile software projects. *The Journal of Systems & Software, 81*(6), 961-971. doi: 10.1016/j.jss.2007.08.020

Cohn, M., & Ford, D., (2003). Introducing an agile process to an organization. *International Journal of Software Engineering and Knowledge Engineering, 36*(6), 74–78. doe:10.1109/MC.2003.1204378

Cooke-Davies, T. J., Crawford, L. H., & Lechler, T. G. (2009). Project management systems: Moving project management from an operational to a strategic discipline. *Project Management Journal, 40*(1), 110-123. doi:10.1002/pmj.20106

Cooper, D. R., & Schindler, P.S., (2014). *Business research methods.* (12th ed.). New York: McGraw-Hill Publisher.

Creswell, J. W. (2014). *Research design: Qualitative, quantitative, and mixed methods approaches* (4th ed.). Thousand Oaks, CA: Sage Publications.

Daniel, D. R. (1961). Management information crisis. *Harvard Business Review, 39,* 111-121.

Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2012). *The Scrum primer: A lightweight guide to the theory and practice of Scrum version 2.0*. Retrieved from http://scrumprimer.com

Deutskens, E., Ruyter, K. D., Wetzels, M., & Oosterveld, P. (2004). Response rate and response quality of internet-based surveys: An experimental study. *Marketing Letters, 15*(1), 21-36. doi:10.1023/B: MARK.0000021968.86465.00

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *The Journal of Systems and Software, 119*, 87-108. doi: 10.1016/j.jss.2016.06.013

Dyba, T., & Dingsoyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology, 50*(9), 833-859. doi: 10.1016/j.infsof.2008.01.006

Dyba, T., & Dingsoyr, T. (2009). What do we know about agile software development? *IEEE Software, 26*(5), 6-9. doi: http://dx.doi.org.library.capella.edu/10.1109/MS.2009.145

Fernandez, D., & Fernandez, J. (2008). Agile project management – Agilism versus traditional approaches. *The Journal of Computer Information Systems, 49*(2), 10-17. Retrieved from http://www.tandfonline.com/doi/abs/10.1080/08874417.2009.11646044

Florentine, S. (2015, July). Top 10 project management certifications. *CIO from IDG*. Retrieved from http://www.cio.com/article/2945413/certifications/top-10-project-management-certifications.html

FluidSurveys (2017). *FluidSurveys home page*. Retrieved from http://fluidsurveys.com

G*Power. (2016). *G*Power 3.1.6 download*. Retrieved from http://www.g-power.com/about.com

Gandomani, T. J., Zulzalil, H., Abdul Ghani, A. A., Md Sultan, A. B., & Sharif, K. Y. (2014). Exploring facilitators of transition and adoption to agile methods: A grounded theory study. *Journal of Software, 9*(7), 1666-1678. doi:10.4304/jsw.9.7.1666-1678

Gantt.com. (2016). *What is a Gantt chart? Gantt chart information, history and software*. Retrieved from http://www.gantt.com

Ghani, I., Bello, M., & Bagiwa, I. L. (2015). Survey-based analysis of agile adoption on performances of IT organizations. *Journal of Korean Society for Internet Information, 16*(5), 87-92. doi:10.7472/jksii.2015.16.5.87

Gloeckner, G. W., Gliner, J. A., Tochterman, S. M., & Morgan, G. A. (2001). Assessing validity and reliability for data collection instruments. In E. I. Farmer & J. W. Rojewski (Eds.), *Research Pathways: Writing Professional Papers, Theses, and Dissertations in Workforce Education* (pp. 223-245). Lanham, MD: University Press of America.

Gonçalves, E., & Lopes, E. (2014). Implementing Scrum as an IT project management agile methodology in a large scale institution. *Proceedings of the 13th European Conference on Research Methods for Business and Management*, UK, 461-470. Retrieved from http://5.202.51.177:2626/handle/Hannan/11826#sthash.D5WMw1Ji.dpbs

Grant, B., & Kelly, K. (2009). *The evolution of project management*. Retrieved from http://www.xchangor.com/history_of_pm.htm

Gupta, S. (n.d.). *9 Principles (building blocks of DSDM – agile. Quotium.com*. Retrieved from http://www.quotium.com/performance/9-principles-building-blocks-dsdm-agile/

Hanisch, J., & Corbitt, B. (2007). Impediments to requirements engineering during global software development. *European Journal of Information Systems, 16*(6), 793-805. doi: 10.1057/palgrave.ejis.3000723

Hastie, S., & Wojewoda, S. (2015). *Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch*. Retrieved from http://www.infoq.com/articles/standish-chaos-2015

Heusser, M. (2015). Introducing the scales agile framework. *CIO from IDG.* Retrieved from http://www.cio.com/article/2936942/enterprise-software/introducing-the-scaled-agile-framework.html?page=2

Highsmith, J. (2002). What is agile software development? *CROSSTALK: The Journal of Defense Software Engineering, 15*(10)*,* 4-9. Retrieved from http://agilesweden.com/doc/oct02.pdf

Hoda, R., & Murugesan, L. K. (2016). Multi-level agile project management challenges: A self-organizing team perspective. *The Journal of Systems & Software, 117*, 245-257. doi: 10.1016/j.jss.2016.02.049

IBM. (1998). *Rational unified process best practices for software development teams. Rational Software white paper*. Retrieved from https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

IBM (n.d.). *IBM SPSS Software*. Retrieved from https://www.ibm.com/analytics/us/en/technology/spss/

Intelliware (2016). *Challenges of agile adoption*. Retrieved from http://www.intelliware.com/challenges-of-agile-adoption/

Interventions.org (2015). *PERT/CPM for project scheduling & management.* Retrieved from http://www.interventions.org/pertcpm/

Jeremiah, J. (2015). *Survey: Is agile the new norm?* Retrieved from http://techbeacon.com/survey-agile-new-norm

Joshi, A., Kale, S., Chandel, S., & Pal, D. (2015). Likert scale: Explored and explained. *British Journal of Applied Science & Technology, 7*(4), 396-403. doi:10.9734/BJAST/2015/14975

Kaleshovska, N., Josimovski, S., Pulevska-Ivanovska, L., Postolov, K., & Janevski, Z. (2015). The contribution of Scrum in managing successful software development projects. *Economic Development, 17*(1/2), 175-194.

Khalil, C., & Khalil, S. (2016). A governance framework for adopting agile methodologies. *International Journal of e-Education, e-Business, e-Management and e-Learning, 6*(2), 111. doi:10.17706/ijeeee.2016.6.2.111-119

Khorrami Rad, N. (2014, February). PMP vs. PRINCE2 certificates. *Project Smart*. Retrieved from https://www.projectsmart.co.uk/pmp-vs-prince2-certificates.php

Kozak-Holland, M. & Procter, C., (2014, February). Florence Duomo project (1420-1436): Learning best project management practice from history. *International Journal of Project management (0263-7863), 32*(2), p. 242.

Laanti, M., Salo, O., & Abrahamsson, P. (2011). Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. *Information and Software Technology, 53*(3), 276-290. doi: 10.1016/j.infsof.2010.11.010

Laerd Statistics (2015). *Multiple regression using SPSS statistics*. Retrieved from https://statistics.laerd.com/

Lech, P. (2013). Time, budget, and functionality? IT project success criteria revised. *Information Systems Management, 30*(3), 263-275. doi:10.1080/10580530.2013.794658

Libweb.com. (n.d.). *Module 9: introduction to research*. Retrieved from http://libweb.surrey.ac.uk/library/skills/Introduction%20to%20Research%20and%20Managing%20Information%20Leicester/page_51.htm

Lindstrom, L., & Jeffries, R. (2004). Extreme programming and agile software development methodologies. *Information Systems Management, 21*(3), 41-52. Retrieved from https://doi.org/10.1201/1078/44432.21.3.20040601/82476.7

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., & Kiefer, D. (2004). Agile software development in large organizations. *International Journal of Software Engineering and Knowledge Engineering, 37*(12), 26–34. doe:10.1109/MC.2004.231

Manifesto for Agile Software Development (2001). Retrieved from http://www.agilemanifesto.org

Matalonga, S., Solari, M., & Matturro, G. (2013). Factors affecting distributed agile projects: A systematic review. *International Journal of Software Engineering and Knowledge Engineering, 23*(9), 1289-1301. doi:10.1142/S021819401350040X

McLaughlin, M. (2016). *What is agile methodology?* Retrieved from https://www.versionone.com/agile-101/agile-methodologies/

Millhollan, C., & Kaarst-Brown, M. (2016). Lessons for IT project manager efficacy. A review of the literature associated with project success. *Project Management Journal, 47*(5), 89-106. Retrieved from https://www.pmi.org/learning/library/it-project-manager-efficacy-literature-review-10276

Misra, S. C., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *The Journal of Systems & Software, 82*(11), 1869-1890. doi: 10.1016/j.jss.2009.05.052

Mitchell, M. L., & Jolley, J. M. (2010). *Research design explained* (7th ed.). Australia: Wadsworth.

Muller, R., & Jugdev, K. (2012). Critical success factors in projects. *International Journal of Managing Projects in Business, 5*(4), 757-775. doi: https://doi.org/10.1108/17538371211269040

Optimus Information.com (2015). *Agile software development project.* Retrieved from http://www.optimusinfo.com/traditional-vs-agile-software-development/

Osborne, J.W. & Waters, E. (2002). Four assumptions of multiple regression that researchers should always test. *Practical Assessment, Research & Evaluation 8(2)*. Retrieved from http://pareonline.net/getvn.asp?n=2&v=8

Papatheocharous, E., & Andreou, A. S. (2014). Empirical evidence and state of practice of software agile teams. *Journal of Software: Evolution & Process, 26*(9), 855-866. doi:10.1002/smr.1664

Parke, C. (2013). *Module 7: Evaluating model assumptions for multiple regression analysis*. Thousand Oaks: SAGE Publications, Inc. doi:10.4135/9781506335148.n7

Pew Research Center. (2017). *Internet surveys*. Retrieved from http://www.people-press.org/methodology/collecting-survey-data/internet-surveys/

Project Management Institute. (2013). *A guide to the project management body of knowledge* (5th ed.). Newton Square, PA: Project Management Institute, Inc.

Project Management Institute (2017). *Learn about PMI*. Retrieved from https://www.pmi.org/about/learn-about-pmi/what-is-project-management

Roberts, P., Priest, H., & Traynor, M. (2006). Reliability and validity in research. *Nursing Standard, 20*(44) 41-45. Retrieved from https://www.researchgate.net/file.PostFileLoader.html?id=5870af51b0366d919a6edc51&assetKey=AS%3A447820391030786%401483780109831

Rockart, J. F. (1979). Chief executives define their own data needs. *Harvard Business Review, 57*(2), 81-93. Retrieved from https://hbr.org

Rockart, J. F., & Crescenzi, A. D. (1984). Engaging top management in information technology. *Sloan Management Review, 25*(4), 3. Retrieved from http://www.sims.monash.edu.au/subjects/ims5042/stuff/readings/rockart_crescenzi.pdf

Rongala, A. (2015, November). *Top 10 project management certifications in demand*. Retrieved from https://www.invensislearning.com/blog/top-10-project-management-certifications-in-demand/

Sargut, G., & McGrath, R. G. (2011). *Learning to live with complexity*. Retrieved from https://hbr.org/2011/09/learning-to-live-with-complexity

Saynisch, M. (2010a). Beyond frontiers of traditional project management: An approach to evolutionary, self-organizational principles and the complexity theory—Results of the research program. *Project Management Journal, 41*(2), 21-37. doi:10.1002/pmj.20159

Saynisch, M. (2010b). Mastering complexity and changes in projects, economy, and society via project management second order (PM-2). *Project Management Journal, 41*(5), 4-20. doi:10.1002/pmj.20167

Scaled Agile. (n.d.) *Scaled agile*. Retrieved from https://www.scaledagile.com/enterprise-solutions/how-to-safe/go-safe/

Schwaber, K., & Sutherland, J. (2016). *The Scrum guide: The definitive guide to Scrum: The rules of the game*. Retrieved from http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100

Scrum Alliance (2016a). *Glossary of Scrum terms*. Retrieved from https://www.scrumalliance.org/community/articles/2007/march/glossary-of-scrum-terms#1118

Scrum Alliance (2016b). *Home page*. Retrieved from https://www.scrumalliance.org

Scrum Alliance (2016c). *Learn about Scrum*. Retrieved from https://www.scrumalliance.org/why-scrum

Scrum Alliance (2016d). *The Scrum team*. Retrieved from https://www.scrumalliance.org/why-scrum/scrum-guide

Scrum Alliance Facebook (2017). *Scrum Alliance Facebook: About*. Retrieved from https://www.facebook.com/pg/scrumalliance/about/?ref=page_internal

Scrum.org (n.d.). *What is Scrum?* Retrieved from https://www.scrum.org/Resources/What-is-Scrum

SCRUMstudy (2015). *SCRUMstudy LinkedIn*. Retrieved from http://www.scrumstudy.com/AboutUs/join-linkedin-group

Seltman, H. (2015). *Experimental design and analysis.* Retrieved from http://www.stat.cmu.edu/~hseltman/309/Book/Book.pdf

Senapathi, M., & Srinivasan, A. (2012). Understanding post-adoptive agile usage: An exploratory cross-case analysis. *The Journal of Systems and Software, 85*(6), 1255-1268. doi: 10.1016/j.jss.2012.02.025

Sharp, J. H., & Ryan, S. D. (2011). Global agile team configuration. *Journal of Strategic Innovation and Sustainability, 7*(1), 120-134. Retrieved from http://search.proquest.com.library.capella.edu/docview/885149967?accountid=27965

Stankovic, D., Nikolic, V., Djordjevic, M., & Cao, D. (2013). A survey study of critical success factors in agile software projects in former Yugoslavia IT companies. *The Journal of Systems and Software, 86*(6), 1663-1678. doi: 10.1016/j.jss.2013.02.027

SurveyMonkey. (2017). *SurveyMonkey take a tour*. Retrieved from https://www.surveymonkey.com/mp/take-a-tour/

Tabachnick, B. G., & Fidell, L. S. (2003). *Using multivariate statistics* (4th ed.). Boston: Allyn and Bacon.

Tran, B. (2016). Communication (intercultural and multicultural) at play for cross cultural management within multinational corporations (MNCs). In N. Zakaria, A.N. Abdul-Talib, & N. Osman (Eds.). *Handbook of Research on Impacts of International Business and Political affairs on the global economy (*pp. 62-92). Hershey, PA: IGI Global.

Trochim, W. M., (2006). *The research methods knowledge base* (2nd ed.). Retrieved from http://www.socialresearchmethods.net/kb/index.php

U.S. Department of Health & Human Services. (1979). *Belmont Report*. Retrieved from http://www.hhs.gov/ohrp/humansubjects/guidance/belmont.html

VersionOne (2016a). *The 10th annual state of agile report*. Retrieved from https://versionone.com/pdf/VersionOne-10th-Annual-State-of-Agile-Report.pdf

VersionOne (2016b). *What is agile methodology?* Retrieved from https://www.versionone.com/agile-101/agile-methodologies/

Willits, F. K., Theodori, G. L., & Luloff, A. (2016). Another look at Likert scales. *Journal of Rural Social Sciences, 31*(3), 126.

Yale University (n.d.). *Multiple linear regression*. Retrieved from http://www.stat.yale.edu/Courses/1997-98/101/linmult.htm

APPENDIX A. STATEMENT OF ORIGINAL WORK

Academic Honesty Policy

Capella University's Academic Honesty Policy (3.01.01) holds learners accountable for the integrity of work they submit, which includes but is not limited to discussion postings, assignments, comprehensive exams, and the dissertation or capstone project.

Established in the Policy are the expectations for original work, rationale for the policy, definition of terms that pertain to academic honesty and original work, and disciplinary consequences of academic dishonesty. Also stated in the Policy is the expectation that learners will follow APA rules for citing another person's ideas or works.

The following standards for original work and definition of *plagiarism* are discussed in the Policy:

> Learners are expected to be the sole authors of their work and to acknowledge the authorship of others' work through proper citation and reference. Use of another person's ideas, including another learner's, without proper reference or citation constitutes plagiarism and academic dishonesty and is prohibited conduct. (p. 1)
>
> Plagiarism is one example of academic dishonesty. Plagiarism is presenting someone else's ideas or work as your own. Plagiarism also includes copying verbatim or rephrasing ideas without properly acknowledging the source by author, date, and publication medium. (p. 2)

Capella University's Research Misconduct Policy (3.03.06) holds learners accountable for research integrity. What constitutes research misconduct is discussed in the Policy:

> Research misconduct includes but is not limited to falsification, fabrication, plagiarism, misappropriation, or other practices that seriously deviate from those that are commonly accepted within the academic community for proposing, conducting, or reviewing research, or in reporting research results. (p. 1)

Learners failing to abide by these policies are subject to consequences, including but not limited to dismissal or revocation of the degree.

Statement of Original Work and Signature

I have read, understood, and abided by Capella University's Academic Honesty Policy (3.01.01) and Research Misconduct Policy (3.03.06), including Policy Statements, Rationale, and Definitions.

I attest that this dissertation or capstone project is my own work. Where I have used the ideas or words of others, I have paraphrased, summarized, or used direct quotes following the guidelines set forth in the APA *Publication Manual*.

Learner name and date Lorena A Stanberry 10/25/2017