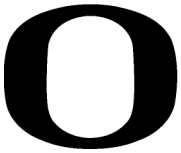
Presented to the Interdisciplinary Studies Program:



UNIVERSITY OF OREGON
APPLIED INFORMATION MANAGEMENT

Applied Information Management
and the Graduate School of the
University of Oregon
in partial fulfillment of the
requirement for the degree of
Master of Science

Best Practices for Large-Scale Agile Transformations

CAPSTONE REPORT

Del Wright
Technology Manager
State Farm Insurance

University of Oregon
Applied Information
Management
Program

Summer 2018

Continuing and Professional
Education
1277 University of Oregon
Eugene, OR 97403-1277
(800) 824-2714

Approved by

_____

Dr. Kara McFall
Director, AIM Program

Best Practices for Large-Scale Agile Transformations

Del Wright

State Farm Insurance

Abstract

Agile software development methods were originally designed for small, individual team projects, but have shown potential benefits for larger projects and organizations as well. Concerns exist, however, regarding scalability and complexity when integrating Agile practices in large-scale organizations. Using a review of literature published since 2011, this paper seeks to inform information technology (IT) leaders of large organizations of the challenges and best practices for implementing large-scale Agile transformations.

Keywords: Agile, software, development, large-scale, transformation, best practices

Table of Contents

Abstract 3

Introduction to the Annotated Bibliography 7

Problem Statement 7

Purpose Statement 11

Research Question 11

Audience Description 12

Search Report 12

Documentation Approach 14

Reference Evaluation Criteria 14

Annotated Bibliography 16

Introduction to the Annotated Bibliography 16

Background and History of Agile Methodologies 16

Challenges of Large-Scale Agile Implementations 22

Best Practices of Large-Scale Agile Implementations 32

Conclusion 49

Background and History of Agile Methodologies 49

Challenges of Large-Scale Agile Implementations 51

Best Practices of Large-Scale Agile Implementations 53

Final Thoughts 55

References 57

Introduction to the Annotated Bibliography

Problem Statement

Large organizations are delivering increasingly advanced business products and services to their customers and stakeholders through complex, innovative, and unique projects and programs (Pawel, 2017). The Project Management Institute (2017) defines a project as "a temporary endeavor undertaken to create a unique product, service, or result" (p. 4) and defines a program as "a group of related projects, subsidiary programs, and program activities managed in a coordinated manner to obtain benefits not available from managing them individually" (p. 11). The top issues and challenges that contemporary organizational projects and programs face include the increased need for transparency in project planning; reduced time-to-market; improved communication with customers and project teams; development of the right organizational culture; and increased predictability of customers' deliveries, project efficiency, effectiveness, innovation, and development speed (Pawel, 2017).

Software development projects and programs face these same challenges, as well as the threat of potential failure (Henriksen & Pedersen, 2017). The number of failing software projects each year is high, as only 9 percent, 16 percent, and 28 percent of them are considered successful in large, medium, and small companies, respectively (Henriksen & Pedersen, 2017). According to Charette (2005), some of the main causes of software project failure are unrealistic or unarticulated project goals; inaccurate estimates of needed resources; badly defined system requirements; poor reporting of the project's status; unmanaged risks; poor communication among customers, developers, and users; inability to handle the project's complexity; sloppy development practices; and poor project management (p. 45).

One methodology that holds promise in addressing these challenges is Agile development (Campanelli & Parreiras, 2015). The Agile development methodology was formed in 2001 when a group of software developers met at a ski resort in Utah to develop the Manifesto for Agile Software Development (Beck et al., 2001). Rather than focus on the differences and competitive advantages of the many software development methodologies that existed at the time, the common interests and philosophies the 17 attendees discussed effectively rocked the software industry when they forged a new approach to software development, in the process coining the term "Agile software development" (Williams, 2012, pp. 71-72). According to the *Agile Manifesto*, Beck et al. (2001) noted that as they uncovered better ways of developing software, they came to value:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan (para. 1).

The authors of the *Agile Manifesto* identified as their highest priority the need to satisfy the customer by delivering software early and continuously (Beck et al., 2001). Rather than avoiding requirements changes or bogging down the development process with complex change processes, Beck et al. (2001) encouraged – even welcomed – changing requirements, even late in the development cycle, arguing that the ability to harness change provides a competitive advantage for the customer. They encouraged alignment between the efforts of the development team and the business by having both parties work together daily throughout the project (Beck et al., 2001).

With origins in software development, Agile methods have begun revolutionizing the way work efforts are organized and executed, and have gained increased attention in the general field of project management (Stettina & Hörz, 2015). Agile software development principles have not changed substantially since the development of the Agile Manifesto in 2001; Dikert, Paasivaara, and Lassenius (2016) define Agile software development as "a set of iterative and incremental software engineering methods that are based on 'Agile philosophy' captured in the Agile Manifesto" (p. 88). Agile development includes methods in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams (Gregg, Scharadin, & Clements, 2016). Compared to plan-based, predictive project methodologies such as Waterfall, Agile is fast and flexible due to frequent feedback loops, close interaction with customers, and iterative reviews (Stettina & Hörz, 2015).

Agile development presents a viable option to achieve quality products and services, budget control, alignment with business strategy, and frequent and continuous delivery of business value (Campanelli & Parreiras, 2015). According to Abbas, Gravell, and Wills (2008), Agile methods are a reaction to the increasing change in the business and technology environment where traditional approaches simply cannot cope with frequently changing requirements that occur over the life of a project. Agile software development projects have been shown to allow for greater adaptability with changing requirements, focus more on customer needs, produce faster development cycles, and deliver functionality on a more frequent basis (Barlow et al., 2011; Hobbs & Petit, 2017). Across the software industry, there is a common view that using Agile on a software project will result in a greater likelihood of success (Henriksen & Arne R. Pedersen, 2017). In their research, Serrador and Pinto (2015) found that the level of Agile use in a project has a statistically significant correlation to a project's

efficiency, stakeholder satisfaction, and overall project success. Serrador and Pinto (2015) define project efficiency as meeting cost, time, and scope goals and stakeholder satisfaction as meeting the expectations of project stakeholders who are the best judges of overall success.

Agile development methods were originally designed for small, individual team projects, but have shown potential benefits for larger projects and organizations as well (Dikert, et al., 2016). While many reasons exist for large organizations to adopt Agile methods, the most important are accelerating time-to-market, managing changing priorities, improving alignment to business goals, enhancing productivity, increasing software quality, and boosting customer satisfaction (Gandomani & Nafchi, 2016). Other potential benefits for large organizations that successfully implement Agile development methods include improvements in the areas of management and organization, increased business performance, maximized shareholder value, and increased competitive advantage in a dynamic and unpredictable marketplace (Pawel, 2017). The potential benefits of Agile methodologies for large organizations are promising, but challenges in implementation remain (Turetken, Stojanov, & Trienekens, 2016).

Since its introduction in 2001, Agile methods have gained wide acceptance; however, concerns regarding the scalability and integration of Agile practices in large-scale organizations continue (Turetken et al., 2016). In this context, large-scale organizations are defined as software development organizations with 50 or more people or at least six teams (Dikert et al., 2016). Implementing Agile methods in large organizations is more difficult than in small ones in part because size brings higher organizational inertia, which slows down organizational change, and successful adoption requires change of the entire organizational culture (Paasivaara & Lassenius, 2016). Changing the culture of a large organization is challenging because it requires the reevaluation and adjustment of communication paths, human resource policies, and management

approaches to better align with the team-based nature of Agile development (Papadopoulos, 2015). Large organizations also have more dependencies between projects and teams, which increases the need for formal documentation and inter-team coordination, thus reducing agility (Paasivaara & Lassenius, 2016). Other challenges in implementing Agile development methodologies in large-scale organizations include the absence of business and customer involvement, existing IT landscapes that are large and complex, the lack of team collocation and collaboration, and the need to integrate Agile projects with existing processes and project management methodologies (Hobbs & Petit, 2017; van Waardenburg & van Vliet, 2013).

Despite concerns about increased complexity and the need for coordination, there is an industry trend towards adopting Agile methodologies at large scale (Dikert et al., 2016). As Agile adoption increases in popularity, the question large organizations ask themselves shifts away from why to adopt these practices, to how to successfully adopt and scale them (Turetken et al., 2016).

Purpose Statement

The purpose of this annotated bibliography is to present literature that addresses the problem of implementing Agile software development methodologies in large-scale, multi-team organizations. This study is significant to information technology (IT) leaders of large-scale organizations who are considering adopting Agile practices beyond that of small, single teams by examining the history and benefits of Agile software development along with the challenges and best practices of implementing it at scale.

Research Question

Main Question. What are best practices for the successful adoption and implementation of Agile development methodologies within large-scale organizations?

Sub Questions. What formal or commercial frameworks have been developed to execute upon large-scale Agile practices? What factors must be considered when choosing the right framework for an organization?

Audience Description

This study is meant to inform chief information officers, vice presidents of technology, IT directors, IT managers, and other IT leaders of large organizations who are, or are considering, executing large-scale Agile transformations within their organizations. While the potential benefits of Agile have made this methodology attractive beyond the context of small, single-team projects, the methodology is more difficult to implement at a larger scale (Dikert et al., 2016). By compiling research focused on the challenges of implementing Agile in large enterprises, those members of IT leadership will be better informed as they develop their strategic and tactical plans for similar organizational transformations. Leveraging lessons learned from organizations who pursued Agile implementations before them will enable these leaders and other change agents to avoid pitfalls that come with this type of organizational change, while also identifying key factors and best practices that are common for success.

Search Report

Search Strategy. I used the University of Oregon Library system (UO Libraries) and its collection of online databases as the primary tool to search for and locate literature on the topic of Agile implementations in large organizations. I conducted initial searches using key terms and phrases including *agile project management, scaled agile,* and *large scale Agile.* I later used an expanded set of key words after identifying related terms and tags embedded within the returned literature. If I was able to locate a potentially relevant article via the UO Libraries but unable to

access the full text, I utilized Google Scholar as an alternate means of retrieving the article by searching by its title.

Key terms. I performed a search of online databases and search engines using the following key terms and phrases:

- agile project management;
- large scale agile;
- large scale scrum;
- scaled agile;
- agile transformation;
- enterprise agile;
- agile scaling;
- agile organization;
- agile practices;
- agile methods;
- agile best practices;
- agile software development;
- agile development methodologies; and
- SAFe.

Search engines and databases. I utilized the following online databases and search engines to locate literature on my topic:

- Academic Search Premier.
- ACM Digital Library.
- Business Source Complete.

- Elsevier.
- IEEE Xplore.
- JSTOR.
- ScienceDirect.
- Wiley Online Library.
- Google Scholar.

Documentation Approach

My approach for documenting references included the use of multiple electronic tools. First, I used Zotero to collect key metadata information about the article and its source. I initially stored this information in a folder that corresponded to one of three categories related to my topic. I then exported the bibliographic citation using American Psychological Association (APA) sixth edition formatting to a local document on my personal computer, where I also organized the references by related category. I added the article's abstract, key word(s) used for searching, and database where retrieved to this file. I downloaded a portable document format (PDF) copy of each article and stored each copy in a folder that corresponded to its applicable category.

Reference Evaluation Criteria

When evaluating the sources used within this research study, I relied upon the five criteria recommended by the Center for Public Issues Education (2014): *authority*, *timeliness*, *quality*, *relevancy*, and (lack of) *bias*.

Authority. I established authority by selecting references that were published in peer-reviewed, scholarly journals or papers presented at recognized conference proceedings. Whenever possible, I reviewed the author's biographical information to verify professional

credentials and education history. I also considered how frequently the article was cited by other authors, as those with frequent citations in peer-reviewed journals are established authorities.

Timeliness. To ensure timeliness of the literature collected, I excluded sources published prior to 2011 except for those information sources required to establish background and history of the research topic. In those limited cases, I included sources published since 2001.

Quality. I verified that sources I selected adhered to proper grammar, spelling, and punctuation, and that information was clear and well-organized.

Relevancy. I established relevancy by ensuring that literature I selected pertained directly to the topic of study, and was obtained from scholarly sources versus popular ones.

Bias. To ensure that the literature I selected was free from bias, I only selected articles whose conclusions were supported by multiple credible and cited sources. I also avoided sources from vendors whose purpose was to sell a given product or service.

Annotated Bibliography

Introduction to the Annotated Bibliography

This annotated bibliography presents 15 references that focus on topics to aid in the successful adoption of Agile software development methodologies within large organizations and are sorted into one of three categories: (a) background and history of Agile methodologies, (b) challenges of large-scale Agile implementations, and (c) best practices of large-scale Agile implementations. Each reference contains a full bibliographic citation, its published abstract, and summary that describes the relevance of the source to this study. All ideas and opinions expressed in the summaries are those of the source reference's author(s).

Background and History of Agile Methodologies

Abbas, N., Gravell, A. M., & Wills, G. B. (2008). Historical roots of Agile methods: Where did "Agile thinking" come from? In *Agile Processes in Software Engineering and Extreme Programming* (pp. 94–103). Heidelberg, Berlin: Springer. https://doi.org/10.1007/978-3-540-68255-4_10

Abstract. The appearance of Agile methods has been the most noticeable change to software process thinking in the last fifteen years, but in fact many of the "Agile ideas" have been around since 70's or even before. Many studies and reviews have been conducted about Agile methods which ascribe their emergence as a reaction against traditional methods. In this paper, we argue that although Agile methods are new as a whole, they have strong roots in the history of software engineering. In addition to the iterative and incremental approaches that have been in use since 1957, people who criticised [*sic*] the traditional methods suggested alternative approaches which were actually Agile ideas such as the response to change, customer involvement, and working software over documentation. The authors of this paper believe that

education about the history of Agile thinking will help to develop better understanding as well as promoting the use of Agile methods. We therefore present and discuss the reasons behind the development and introduction of Agile methods, as a reaction to traditional methods, as a result of people's experience, and in particular focusing on reusing ideas from history.

Summary. The authors of this paper argue that while Agile software development has been most noticeable since the publishing of the *Agile Manifesto* in 2001, 'Agile' ideas and practices have been around for much longer. The authors seek to provide information on the origin of Agile methods and thinking, along with where and how Agile methods were first introduced. The authors discuss Agile methods as being a reaction to the dissatisfaction of software developers with traditional methods, how Agile methodologies emerged from the reuse of ideas in history, and how people's own development experiences influenced Agile ideas.

The authors state that the emergence of Agile methods is a reaction to the bureaucracy of traditional software development methods and an increasing change in the complexity of the business environment. They argue that traditional methods could not cope with turbulent business and technology changes as they are built around the idea that it is possible to anticipate a complete set of business and system requirements early in the development lifecycle. The authors note that most changes in requirements and technology occur throughout a project's life span.

The authors present several examples of how people long ago were embracing Agile ideas and practice as the most successful way of building software. For example, iterative and incremental development (IID) was embraced by NASA in their 1961-63 Project Mercury, where they leveraged half-day build iterations, continuous code integration, and the Extreme Programming (XP) practice of test-driven development where tests were planned and written

first, then code written to pass the tests. The authors note that in 1970, Winston Royce suggested the use of a development pilot and the early and regular involvement of the customer throughout the development lifecycle as methods to help mitigate risk that is found in sequential development approaches. The authors describe how the *Evo*lutionary Project Management (EVO) method introduced by Tom Gilb in 1985 was based on the principles of (a) deliver something to the real end-user, (b) measure the added value to the user in all critical dimensions, and (c) adjust both design and objectives based on observed realities. Lastly, James Martin's 1991 Rapid Application Development (RAD) and its fundamentals of quick delivery; iterative development; small, highly-skilled teams; and heavy user involvement are noted as being consistent with the theme of Agile methods.

The authors also describe how people's own experiences contributed to the foundations of what is known as Agile development. As they write, most people involved in developing the *Agile Manifesto* had experience with their own well-defined methods, such as Ken Schwaber with Scrum, Kent Beck with XP, and Alistair Cockburn with Crystal.

This paper is relevant to the study as it provides historical evidence that Agile ideas and software development methods have existed long before the *Agile Manifesto* was written. It suggests that Agile software methods are not a recent fad, and instead an evolution of approaches that have been successful since as early as 1957.

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M.,...Thomas, D. (2001). Manifesto for Agile software development. Retrieved from http://www.agilemanifesto.org/

Abstract. Note: This abstract was written by the author of this annotated bibliography in the absence of a published abstract in the source. On February 11-13, 2001, at The Lodge at

Snowbird ski resort in the Wasatch mountains of Utah, 17 people met to talk, ski, relax, and try to find common ground. 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 convened. What emerged from this meeting was the *Manifesto for Agile Software Development*, signed by all participants.

Summary. The *Manifesto for Agile Software Development* was developed by 17 software developers who were seeking a viable alternative to software development processes that were document-driven and cumbersome. The authors describe the four values of the *Agile Manifesto:*

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan (para. 1).

In addition, the authors defined 12 principles behind the *Agile Manifesto* that make Agile methods and those who embrace them unique (Beck et al., 2001):

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.

- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The *Agile Manifesto* is significant to this study because it is the seminal source for Agile software development methodologies and describes the specific values and principles on which modern day Agile software development is based. Software development teams, project management methodologies, and other organizational units that identify as being *Agile* generally refer back to the four values and 12 principles that make up the *Agile Manifesto.*

Williams, L. (2012). What Agile teams think of Agile principles. *Communications of the ACM*, *55*(4), 71–76. https://doi.org/10.1145/2133806.2133823

Abstract. The article focuses on agile software development and its use by practicing software engineers. It talks about the state of software development methodology in the 1990s which led to independent consultants developing agile software development tools and mentions that in 2001 17 software engineers codified the 12 original agile principles into the Agile

Manifesto. It analyzes data from surveys of software programmers to determine how many of them used agile principles and their thoughts on the value of the principles. It mentions the introduction of lean software development kanban practices.

Summary. By surveying 326 practitioners, the author of this article sought to understand how well the original 12 principles of the *Agile Manifesto* published in 2001 still reflect what is valued by software developers and teams practicing Agile methodologies almost a decade later. The first set of questions focused on the original principles by asking 326 respondents to rate on a scale of 1-5 (1 = not important; 5 = essential) how important each principle was for Agile teams in 2010. The author noted that 11 of the 12 principles had a mean score of 4.1, with the only one scoring below that threshold being Principle 11 – *The best architectures, requirements, and designs emerge from self-organizing teams.*

The second set of questions provided a list of 45 software development practices associated with Agile, and asked respondents to identify the essential ones by using the same scoring scale. The results showed that many of the original Agile practices such as continuous integration and short iterations scored highest, while more recent and emergent practices such as Planning Poker, or a practice for estimating team-based effort; Kanban, or a focus on limiting work in progress in lieu of timeboxing through the use of iterations; and stabilization iterations scored lowest. Ultimately, the author of this article argues that while many Agile practices have matured and evolved over time, a high level of support remains for the original principles from the *Agile Manifesto.*

This article is important to this study as it helps to confirm that the principles stated within the *Agile Manifesto*, along with software development practices that coincide with them, are still relevant and essential to Agile teams years after it was first published. A shortcoming to

note, however, is that this article was published in 2012. Several Agile software development practices that were considered emergent at that time may have found greater maturity and adoption, and corresponding survey responses may not accurately represent the same level of present day sentiment.

Challenges of Large-Scale Agile Implementations

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

Abstract. Agile methods have become an appealing alternative for companies striving to improve their performance, but the methods were originally designed for small and individual teams. This creates unique challenges when introducing agile at scale, when development teams must synchronize their activities, and there might be a need to interface with other organizational units. In this paper we present a systematic literature review on how agile methods and lean software development has been adopted at scale, focusing on reported challenges and success factors in the transformation. We conducted a systematic literature review of industrial large-scale agile transformations. Our keyword search found 1875 papers. We included 52 publications describing 42 industrial cases presenting the process of taking large-scale agile development into use. Almost 90% of the included papers were experience reports, indicating a lack of sound academic research on the topic. We identified 35 reported challenges grouped into nine categories, and 29 success factors, grouped into eleven categories. The most salient success factor categories were management support, choosing and customizing the agile model, training and coaching, and mindset and alignment.

Summary. The authors of this paper present a systematic literature review of large-scale Agile transformations using 52 publications that describe cases where Agile processes were applied to large-scale development efforts. They focus their study on answering two research questions: *What challenges have been reported for large-scale Agile transformations?* and *What success factors have been reported for large-scale Agile transformations?*

Based on their research, the authors identified 35 challenges of implementing Agile on large-scale development projects that they grouped into nine categories. These groups include (a) Agile being difficult to implement, (b) integrating non-development functions, (c) change resistance, (d) requirements engineering challenges, (e) hierarchical management and organizational boundaries, (f) coordination challenges in multi-team environments, (g) lack of investment, (h) the emergence of different approaches in a multi-team environment, and (i) quality assurance challenges. In answering their second question, they identified and grouped 29 success factors into the following eleven categories: (a) choosing and customizing an Agile approach; (b) management support; (c) mindset and alignment, such as a concentration on Agile values, embracing Agile communities, and aligning the organization towards a common goal of introducing new development methods ; (d) providing training and coaching; (e) use of piloting; (f) requirements management; (g) team autonomy; (h) commitment to change; (i) communication and transparency; (j) leadership; and (k) engaging people.

Based on their research and analysis, the authors conclude that large-scale Agile is harder to implement than people expect and that large-scale Agile cannot be just implemented off the shelf but instead must be carefully customized to meet organizations' needs. They conclude that understanding Agile values behind Agile practices is important and aligning the whole

organization towards the common goals makes it easier to succeed in large-scale Agile transformations.

The authors identify five related topics that warrant further study: (a) additional case studies on Agile transformations, (b) Agile scaling practices, (c) Agile scaling frameworks, (d) enterprise-level Agile, and (e) surveys on challenges and success factors.

This article is important to this study because the authors present findings from their systematic literature review of 52 papers describing the most reported challenges and success factors occurring during large-scale Agile transformations.

Hobbs, B., & Petit, Y. (2017). Agile methods on large projects in large organizations. *Project Management Journal, 48*(3), 3-19. Retrieved from https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/pmj/2017-june-july.pdf#page=4

Abstract. Agile methods have taken software development by storm but have been primarily applied to projects in what is referred to as the "agile sweet spot," which consists of small collocated teams working on small, non-critical, green field, in-house software projects with stable architectures and simple governance rules. These methods are being used more and more on large projects, but little documentation is available in the academic literature. This article investigates the adoption and adaptation of agile methods for use on large projects in large organizations. The empirical study is based first on case studies, followed by a survey to validate and enrich the case study results. The results are somewhat paradoxical in that some features are common to almost all observations, whereas others show extreme variability. The common features include use of Scrum methodology and agile coaches, as well as the non-respect of the agile principle of emergent architecture.

Summary. This article investigates the adoption and adaptation of Agile methods for use on large projects in large organizations. The authors seek to answer two research questions. The first question is explored at the project level: *What challenges are encountered when applying Agile methods to large multi-team software projects and what practices have been developed to alleviate those challenges?* The second question is investigated at the organizational level: *How does the context of large, complex organizations affect the adaptation and adoption of Agile methods and vice versa*? The research was conducted via an exploratory, mixed-methods approach. First, qualitative case studies were executed by conducting interviews and analyses of company documentation across 12 projects in six large organizations. This effort was followed by a survey to confirm and enrich the results of the case studies.

The results of the study showed that at the project level, larger efforts are faced with three interrelated organizational challenges: (a) coordination of multiple development teams, (b) organization of a greater number of specialists outside of the development teams, and (c) integration with other systems. Challenges also exist with the interaction between the project and the organization. Implementing Agile methods in a large organization with well-established traditional methods results in significant organizational change. Citing Ivari and Iivari (2011), the authors assert that conflicts often exist between large, traditional organizations and Agile principles as they relate to structures, processes, and culture. The authors provide as examples role definitions, project approval processes that require parameters to be well-defined in advance, and a change in management style from command and control to servant leadership.

According to the authors, another big challenge in implementing Agile in large organizations is the relationship between the project and the client organization where business unit managers are, on average, less knowledgeable and supportive of Agile methods. The authors

note that the product owner on Agile projects is generally represented by a member of business who has the knowledge, availability, and authority to make changes to the product backlog and prioritizes the work executed by the development team. The authors find that on large-scale projects, the reality is that product owners do not always possess these characteristics and that the lack of understanding of this role is the biggest challenge for them.

This article is significant to this study as it details specific challenges that must be overcome by large software projects in large organizations wishing to adopt Agile methods.

Paasivaara, M., & Lassenius, C. (2016). Scaling Scrum in a large globally distributed organization: A case study. In *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)* (pp. 74–83). https://doi.org/10.1109/ICGSE.2016.34

Abstract. We present a case study on scaling Scrum in a large globally distributed software development project at Nokia, a global telecommunications company. We discuss how the case project scaled Scrum while growing from two collocated Scrum teams to 20 teams located in four countries and employing a total of 170 persons. Moreover, we report scaling challenges the case project faced during this 2.5 year journey. We gathered data by 19 semi-structured interviews of project personnel from two sites, interviewees comprising different roles including managers, architects, product owners, developers and testers. The project was highly successful from the business point of view, as agile enabled fast response to customer requirements. However, the project faced significant challenges in scaling Scrum despite attempts at applying the Large-scale Scrum (LeSS) framework. The organization experimented with different ways of implementing scaling practices like implementing common sprint planning meetings, Scrum-of-Scrums meetings, common sprint demos and common

retrospectives, as well as scaling the Product Owner role. We conclude the paper by reflecting on the scaling approach used in the case organization in contrast to the LeSS framework.

Summary. The authors of this paper aim to fill a gap that exists in the literature by presenting empirical research on how commercially popular Agile scaling frameworks such as Large-scale Scrum (LeSS) work in practice, challenges with their implementations, and strategies to overcome those challenges. A case study research design was undertaken using Nokia and its adoption of the LeSS framework. The authors identified five scaling practices used by Nokia for coordinating between multiple teams: (a) use of area product owners, (b) common sprint planning, (c) Scrum of Scrum (SoS) meetings, (d) common sprint demos, and (e) common retrospective. Within most of these practices, however, the authors found issues and challenges as the teams adopted them. For example, the authors found that many team members felt that the common meetings did not provide a good big picture or help in coordination with the teams, and that many did not understand their purpose and considered them a waste of time.

The authors identify four main pain points as they examined the project from the outside: (a) the Agile mindset was partially missing, (b) the product was difficult to divide into reasonable requirement areas, (c) a common view of the Scrum implementation was lacking, and (d) constant market pressure causing time pressure. The authors conclude that both inherent problems in the LeSS framework itself coupled with a poor implementation of it contributed to the problems Nokia experienced. The complexity and significant interdependencies of the product made it a poor match for LeSS, and the project and its members did not succeed in attaining a real ‘Agile mindset’ by adopting important practices prescribed by the framework. As it pertains to the LeSS framework itself, the researchers conclude that there are limits to its

applicability, and that very complex projects with many technical interdependencies might require other scaling approaches.

This article is significant to the study because it describes challenges that a large project and organization faced as they implemented a framework designed specifically to scale Agile practices. It highlights the fact that an Agile scaling framework and associated best practices should not be a considered a silver bullet. Instead, inherent challenges may exist within the framework that may not be suitable for every project or organization.

Saeeda, H., Arif, F., Minhas, N. M., & Humayun, M. (2015). Agile scalability for large scale projects: Lessons learned. *Journal of Software, 10*(7), 893-903. https://doi.org/10.17706/jsw.10.7.893-903

Abstract. In modern well-known approaches, “agile” has emerged as the leading approach in software industry for the development of the software projects. With different innovative shapes agile is applicable for handling the issues regarding cost, time, continuously change environment and requirements. Agile has proved to be successful in the small and medium size project, however, it has several limitations when applied on large size projects. The aim of this study is to analyze agile approaches in detail, finding its success stories in small and medium size projects and highlighting its limitations for large size projects. This study will identify the current research problem of the agile scalability for large size projects by giving a detail literature review of the identified problem and will synthesize the existing work for covering the identified problem in the agile scalability. Based on it, we can judge the limitations of agile scalability for large size projects and can think of some remedial approach for overcoming these limitations in future.

Summary. The authors' stated purpose for this study is to review literature on Agile scalability and identify limitations faced by organizations implementing Agile methods on large size projects. Citing Petersen and Wohlin (2009), the authors state that using Agile and incremental techniques in large-scale software development projects lead to challenges such as increased maintenance efforts due to the increase in the number of releases, the complexities and overhead associated with coordination and communication between multiple teams, and difficulties in creating and maintaining requirements priorities lists. Other challenges they identified include concerns from developers about the need to attend numerous meetings resulting from scaling Agile, the loss of focus on the bigger picture, lack of a structured schedule usually found in traditional methods, and the additional complexities found with coordinating between Agile and non-Agile teams within the organization. The authors conclude that while large-scale software projects are popular within the IT industry, they also possess more complexities and challenges such as higher risk and unpredictability.

This paper is relevant to this study because it identifies several issues and challenges organizations face when attempting to implement Agile practices in large-scale projects.

van Waardenburg, G., & van Vliet, H. (2013). When Agile meets the enterprise. *Information and Software Technology, 55*(12), 2154–2171. https://doi.org/10.1016/j.infsof.2013.07.012

Abstract. *Context*: While renowned agile methods like XP and Scrum were initially intended for projects with small teams, traditional enterprise environments, i.e. environments where plan-driven development is prevalent, have also become attracted by the promises of a faster time to market through agility. Agile software development methods emphasize lightweight software development. Projects within enterprise environments, however, are typically confronted with a large and complex IT landscape, where mission-critical information

is at play whose criticality requires prudence regarding design and development. In many an organization, both approaches are used simultaneously. *Objective*: Find out which challenges the co-existence of agile methods and plan-driven development brings, and how organizations deal with those challenges. *Method*: We present a grounded theory of the challenges of using agile methods in traditional enterprise environments, based on a Grounded Theory research involving 21 agile practitioners from two large enterprise organizations in the Netherlands. *Results*: We organized the challenges under two factors: Increased landscape complexity and Lack of business involvement. For both factors, we identify successful mitigation strategies. These mitigation strategies concern the communication between the agile and traditional part of the organization, and the timing of that communication. *Conclusion*: Agile practices can coexist with plan-driven development. One should, however, keep in mind the context and take actions to mitigate the challenges incurred.

Summary. The authors of this paper conducted Grounded Theory research with the objectives of identifying the challenges that result in using Agile methods in traditional enterprise environments and the strategies organizations use to mitigate those challenges. The authors interviewed 21 Agile practitioners from two large enterprise organizations in the Netherlands who were executing multi-team projects of similar size and duration. Using the data collected from the interviews, the authors categorized the identified challenges into two core categories: increased IT landscape complexity and lack of business involvement.

The authors found that increased IT landscape complexity was caused by several factors including the execution of concurrent development streams, different process approaches that increased inter-project dependencies, and the use of a separate layer development approach that included using different teams for front-end development of an application versus back-end

development. Resulting issues identified by the authors included problems with inter-team communications; impediments in meeting the products' definition of done, or the minimum set of activities that adds verifiable value to the product; and difficulties in creating change such as when Agile teams are challenged by non-negotiable requirements, scope, and dependencies established by non-Agile components of the organization.

The authors offered mitigation strategies to address the IT landscape complexity including stimulating a common sense of purpose by all parties involved; managing program-level alignment by combining product backlogs into one to help keep track of changes made by concurrently working teams; creating teams with end-to-end representation such as front-end and back-end developers collectively sharing knowledge and solutions; and facilitating the project management discipline by helping monitor the bigger-picture communication and dependencies that exist in multi-team environments, being well-embedded in the organization to escalate issues when necessary, remaining involved in the team process, and maintaining close and active relationships with the product owner, Scrum Master, and development team.

The authors found that lack of business involvement was caused by the use of a centralized IT department, thus creating a gap between business and IT; and projects that were still organized in a plan-driven way where business involvement is concentrated primarily at the start of the project. These factors resulted in problems obtaining requirements from stakeholders, slow reaction to changes from business, problems prioritizing requirements by the business, and limited feedback from the business as features were implemented.

The authors offered mitigation strategies to address the lack of business involvement including changing the mindset of business stakeholders so that they are more aware of and better understand how Agile methods work in order to obtain and prioritize requirements and

provide feedback earlier and more frequently; channeling business knowledge through the product owner to minimize the number of business representatives teams needed on the project; and aligning knowledge and requirements at the business level through collaborative feature demonstrations and stakeholder workshops.

This paper is relevant to this study as it identifies both challenges and mitigation strategies of integrating Agile software development methods with existing plan-driven ones often found in large, traditional enterprise settings.

Best Practices of Large-Scale Agile Implementations

Alqudah, M., & Razali, R. (2016). A review of scaling Agile methods in large software development. *International Journal on Advanced Science, Engineering and Information Technology, 6*, 828. https://doi.org/10.18517/ijaseit.6.6.1374

Abstract. Agile methods such as Dynamic Systems Development Method (DSDM), Extreme Programming (XP), SCRUM, Agile Modeling (AM) and Crystal Clear enable small teams to execute assigned task at their best. However, larger organizations aim at incorporating more Agile methods owing to the fact that its application is prevalently tailored for small teams. The scope in which large firms are interested will extend the original Agile methods to include larger teams, coordination, communication among teams and customers as well as oversight. Determining particular software method is always challenging for software companies especially when considering start-up, small to medium or large enterprises. Most of large organizations develop large-scale projects by teams of teams or teams of teams of teams. Therefore, most recognized Agile methods or first-generation methods such as XP and SCRUM need to be modified before they are employed in large organizations; which is not an easy task. Accomplishing said task would necessitate large organizations to pick and select from the

scaling Agile methods in accommodating a single vision for large and multiple teams. Deciding the right choice requires wholesome understanding of the method including its strengths and weaknesses as well as when and how it makes sense. Therefore, the main aim of this paper is to review the existing literature of the utilized scaling Agile methods by defining, discussing and comparing them. In-depth reviews on the literature were performed to juxtapose the methods in impartial manner. In addition, the content analysis was used to analyse the resultant data. The result indicated that the DAD, LeSS, LeSS huge, SAFe, Spotify, Nexus and RAGE are the adopted scaling Agile methods at large organizations. They seem to be similar but there are discrepancies among them that take the form of team size, training and certification, methods and practices adopted, technical practices required and organizational type.

Summary. The authors state that the purpose of this paper was to review several popular Agile scaling methods, comprehend the roles and practices found within each scaling method, and identify the differences and similarities among these methods. The seven frameworks the authors reviewed in this paper are Disciplined Agile Delivery (DAD), Scaled Agile Framework (SAFe), Large Scale Scrum (LeSS), the Spotify model, the Nexus method, and Recipes for Agile Governance in the Enterprise (RAGE).

DAD primarily expands upon Scrum, but also incorporates other Agile practices such as Lean, Kanban, and XP. To be successful in adopting DAD, the authors state that delivery teams must work closely with enterprise architects, operations engineers, governance personnel, and data management analysts and a high level of technical competency should be possessed by team members since DAD stresses the use of functional and data modeling. Advantages of DAD noted by the authors include the provision of guidance in the areas of architecture and design, the lack of a prescriptive model so teams can chooses processes that best meet their needs, and the

facilitation of the cooperation of multiple teams within the organization. The largest drawback of DAD the authors identified is the sluggish marketplace adoption of it.

SAFe incorporates practices from Scrum, XP, Kanban, Lean, and DevOps. The authors recommend that organizations planning to adopt SAFe should have personnel and teams well-versed in the use of portfolio management tools. The authors identified advantages of SAFe including a prescriptive structure that eases transition to an Agile framework and an increase in team productivity and return on investment. Likewise, the primary drawback they found is the fact that the prescriptive nature may constrain the continuous development of teams.

LeSS is based on Scrum and applied to more than ten teams working together on one product. The authors deemed LeSS as advantageous for scaling Agile because emphasis on Scrum eases adoption and requires less rigorous training. A drawback the authors identified is less incorporation of other Agile methods besides Scrum as compared to other scaling frameworks such as DAD and SAFe.

Spotify utilized diverse methods such as Kanban, Scrum, and Lean based on the needs of squads at the team levels. The authors note that large organizations interested in adopting this model should have a solid background in each method. The main advantage of Spotify according to the authors is the ability for each squad to select its preferred working method, while a disadvantage is a lack of significant training available for organizations wishing to adopt it.

Nexus is based on Scrum that is applied to more than two but less than nine teams all working collectively on a single product. In addition to incorporating Scrum methods, adopters will incorporate prescribed Nexus methods into their working practices and will require specialized training prior to adoption and implementation. The primary advantage of Nexus noted by the authors is the ability to help solve the problem of interdependencies arising from the

works of multiple teams on a large project, while the primary disadvantage is that Nexus is a newer framework and adoption is not widespread.

RAGE focuses on solving problems of large enterprises which use more than one development method such as Scrum, Kanban, or even traditional plan-driven practices. The primary advantage noted by the authors is that RAGE provides flexibility for teams to follow any method they prefer. RAGE is still new, and therefore the authors note that little empirical evidence exists on how successful it is for adopters.

The authors conclude that while the Agile scaling methods they reviewed are all similar in that they are targeted towards solving problems experienced in large projects, many discrepancies still remain between the methods in the form of required team size, training needs, certification requirements, methods and practices, and organizational type. The authors note that further empirical research is required to assess how and when to best utilize each scaling method for large projects and organizations.

This paper is relevant to this study as it describes six frameworks and models generally recognized as most common for scaling Agile methods in large organizations. The authors compare and contrast the frameworks based on recommended team size, training requirements, methods and practices adopted, technical practices required, and organization type.

Barlow, J., Giboney, J., Keith, M., Wilson, D., Schuetzler, R., Lowry, P., & Vance, A. (2011). Overview and guidance on Agile development in large organizations. *Communications of the Association for Information Systems, 29*(1). Retrieved from http://aisel.aisnet.org/cais/vol29/iss1/2

Abstract. A continual debate surrounds the effectiveness of agile software development practices. Some organizations adopt agile practices to become more competitive, improve

processes, and reduce costs. Other organizations are skeptical about whether agile development is beneficial. Large organizations face an additional challenge in integrating agile practices with existing standards and business processes. To examine the effects of agile development practices in large organizations, we review and integrate scientific literature and theory on agile software development. We further organize our theory and observations into a framework with guidelines for large organizations considering agile methodologies. Based on this framework, we present recommendations that suggest ways large organizations with established processes can successfully implement agile practices. Our analysis of the literature and theory provides new insight for researchers of agile software development and assists practitioners in determining how to adopt agile development in their organizations.

Summary. The authors use a literature review and a theory-based framework to assist practitioners in determining how to best adopt Agile software development in their organizations. The authors discuss situations where either Agile or traditional plan-based methods might be better suited, including a recommendation for when to implement an Agile-traditional hybrid method. The authors assert that Agile methods are more effective in projects with small team sizes and highly reciprocal interdependencies, or those interdependencies that exist when two or more parties depend on each other in both directions. The authors argue that plan-based methods work best in projects of any size where interdependencies are sequential and result from the serial nature of the workflow. The authors state that hybrid methods work best in large projects with many reciprocal interdependencies and provide traditional software development teams with the ability to implement some practices of Agile development to compliment the strengths of their existing methods.

The authors conclude that both plan-based and Agile methods are effective ways to develop software, with each possessing strengths and weaknesses. They identify inherent strengths of plan-based methods as increased formal communication, more robust documentation, and more upfront time dedicated to detailed requirements and design while noting that Agile excels in enabling flexibility and adaptability to changing requirements, shorter development cycles, and an increased focus on customer needs. The authors identify weaknesses of plan-based methods as inflexibility with frequently changing requirements and design and of Agile methods as not promoting formal lines of communication in large projects, unknown initial time and resource requirements, lack of well-defined requirements early on, and lack of detailed documentation. They argue that combining the two to create a hybrid methodology will mitigate inherent weaknesses and create a development methodology that is more effective than either one alone. The authors recommend that large, mature organizations use their framework to select the appropriate development methodology and recommend that those facing high uncertainty and reciprocal interdependencies adopt a hybrid approach. Hybrid approaches, they argue, allow large, mature organizations to enjoy the benefits of Agile development in areas where they have not been previously successful.

This article is relevant to this study because it presents a theory-based framework to assist organizational leaders who may be interested in adopting Agile practices in determining the best approach to use based on factors such as project team size, volatility level, and the nature of project interdependencies.

Gandomani, T. J., & Nafchi, M. Z. (2016). The essential prerequisites of Agile transition and adoption: A grounded theory approach. *Journal of Korean Society for Internet Information, 17*(5), 173–184. https://doi.org/10.7472/jksii.2016.17.5.173

Abstract. Prevalence of Agile methods in software companies is increasing dramatically. Software companies need to employ these methods to overcome the inherent challenges of traditional methods. However, transitioning to Agile approach is a topic of debate and there is no unique and well-defined transition model or framework yet. Although some research studies have addressed barriers and strengths behind the successful Agile deployment, it seems that this process still needs to be studied more in depth. The rationale behind this is the socio-technical nature of Agile transition and adoption. Particularly, the challenges and problems that software companies are facing during Agile transition, show that this process in more difficult than expected. Conducting a large-scale research study revealed that Agile transition and adoption process needs to be supported by several critical prerequisites. This study adopted a Ground Theory with the participation of 49 Agile experts from 13 different countries and empirically identified seven transition prerequisites. These prerequisites focus on the different aspects of the transition. The main aim of this paper is proposing these prerequisites and theoretical and practical implication of these prerequisites. Providing these prerequisites before moving to Agile increases chance of success in Agile transition and adoption and leads to fewer challenges during the change process.

Summary. The authors state that the most important Agile transition and adoption challenges are not technological in nature, but rather social and people-oriented barriers. Based on their research, they identified seven essential prequisites for Agile transition that are related to individuals, team organization, and business management: (a) having a convincing reason for change; (b) people buy-in; (c) defining business goals; (d) initial training; (e) pilot project selection; (f) pre-startup assessment; and (g) team set up.

The authors assert that organizations must first have convincing reasons for shifting from traditional plan-based methods to adopting Agile practices, otherwise they are likely to fail. While there are many clear reasons for making the shift, they argue, each organization must ensure that Agile is indeed appropriate and useful for them. Obtaining stakeholder buy-in is also a requirement for successful Agile transformation. Because Agile methods are people-oriented, people involvement and collaboration is essential. Likewise, management support and commitment is critical in the success of the the Agile transfomation process. The results of their reseach indicated that interested and supportive managers can help manage the transformation process and influence others to adapt to new roles, responsibiities, and ways of working.

The authors argue that defined business goals and benefits are critical drivers with an Agile transormation. They note that they have a potentially large impact on the adoption of, transition to, and governance of Agile software development methods. The authors' research indicates that Agile teams need to be trained in order to be aware of the challenges that exist with Agile transformation and to effectively learn their new roles and responsibiliites.

The authors find that a pilot project selection is critical in companies working on large projects. Choosing an appropriate pilot provides an opportunity for assessing the strengths and weaknesses an organization possesses in deploying Agile practices, as does a pre-startup organizational assessment. Lastly, selecting the right team members for a pilot team strongly affects the success of an Agile tranformation. Choosing the appropriate and qualified members helps to reduce the challenges and barriers, as those who buy-in and embrace Agile concepts will be more successful than those who are indifferent. Likewise, leveraging Agile champions and coaches will help faciliate change and persuade others to change as well.

The research presented in this article is significant to this study because it provides seven essential best practices organizations should consider prior to the adoption and implementation of large-scale Agile transformations.

Paasivaara, M., & Lassenius, C. (2014). Communities of practice in a large distributed agile software development organization – Case Ericsson. *Information and Software Technology, 56*(12), 1556–1577. https://doi.org/10.1016/j.infsof.2014.06.008

Abstract. *Context*: Communities of practice—groups of experts who share a common interest or topic and collectively want to deepen their knowledge—can be an important part of a successful lean and agile adoption in particular in large organizations. *Objective*: In this paper, we present a study on how a large organization within Ericsson with 400 persons in 40 Scrum teams at three sites adopted the use of Communities of Practice (CoP) as part of their transformation from a traditional plan-driven organization to lean and agile. *Methods*: We collected data by 52 semi-structured interviews on two sites, and longitudinal non-participant observation of the transformation during over 20 site visits over a period of two years. *Results*: The organization had over 20 CoPs, gathering weekly, bi-weekly or on a need basis. CoPs had several purposes including knowledge sharing and learning, coordination, technical work, and organizational development. Examples of CoPs include Feature Coordination CoPs to coordinate between teams working on the same feature, a Coaching CoP to discuss agile implementation challenges and successes and to help lead the organizational continuous improvement, an end-to-end CoP to remove bottlenecks from the flow, and Developers CoPs to share good development practices. Success factors of well-functioning CoPs include having a good topic, passionate leader, proper agenda, decision making authority, open community, supporting tools, suitable rhythm, and cross-site participation when needed. Organizational support include creating a

supportive atmosphere and providing a suitable infrastructure for CoPs. *Conclusions*: In the case organization, CoPs were initially used to support the agile transformation, and as part of the distributed Scrum implementation. As the transformation progressed, the CoPs also took on the role of supporting continuous organizational improvements. CoPs became a central mechanism behind the success of the large-scale agile implementation in the case organization that helped mitigate some of the most pressing problems of the agile transformation.

Summary. The authors offer communities of practice as one means of facilitating the scaling of software development. They define a community of practice as "a group of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting on an ongoing basis" (p. 1557). They note that in scaling software development, "CoPs have been proposed as a possible solution for functional learning and knowledge sharing between organizationally separate individuals with similar roles" (p. 1557). Using an exploratory case study across a large organization undergoing an Agile transformation, the authors sought to research what kinds of CoPs were created in the organization and how they evolved over time, the characteristics of successful CoPs, how the organization supported the CoPs, and how the different CoPs could be classified.

The authors found that CoPs had three primary roles in the organization: supporting the Agile transformation, being a part of a large-scale Scrum implementation, and supporting continuous improvement. They identified eight characteristics for successful CoPs that included centering a CoP around an interesting topic from which participants could receive concrete benefit; having a passionate leader who can facilitate meetings with a proper agenda; distributing a useful agenda before meetings and soliciting input into the agenda from all participants; decision-making authority on relevant matters; an open and transparent community supporting

the CoP; tools to build organizational memory and transparency; a suitable meeting rhythm; and the ability for distributed organizations to enable cross-site participation to support alignment with the whole organization. The authors also identified three elements contributing towards the building of a supportive atmosphere for CoPs: openness of participation, participation valued by the organization, and the support of managers and coaches in building the CoPs.

The authors found that CoPs can be used effectively as support mechanisms during large-scale Agile transformations as they provide a forum to discuss the transformation, plan continuous improvements, and share knowledge regarding working practices between roles and teams. They also found that CoPs can support scaling Agile to a large and distributed organization and help provide efficient coordination and knowledge sharing between teams and experts in the teams. Further, their findings suggest that large and complex organizational products might be better supported by product-area or feature-based CoPs rather than basic Scrum-of-Scrums.

The research findings in this article are relevant to this study as they describe how utilizing well-defined and effective CoPs contributed to the success of large-scale Agile transformation within a large organization.

Papadopoulos, G. (2015). Moving from traditional to Agile software development methodologies also on large, distributed projects. *Procedia - Social and Behavioral Sciences, 175*, 455–463. https://doi.org/10.1016/j.sbspro.2015.01.1223

Abstract. The challenge that all companies face in a quickly changing business environment is to stay competitive in order to retain and if possible expand their market share. Traditional software development methods are inflexible and fail to respond on aggressive customer requests. In contrast, agile software methodologies provide a set of practices that allow

for quick adaptations matching the modern product development needs. Although the value of the agile methodologies is well proven for small, collocated teams, the research question that this work is addressing refers to the benefits of the agile methodologies on large, distributed projects. With this paper, evidence is provided by the analysis of a case study that agile software development methodologies perform better than traditional methodologies also in large, distributed projects. Improvements are observed on the quality and on the customer perception of the end product, while agile methodologies allow for requirement changes even late in the project. At the same time, building better communication and collaboration in the team as an outcome of following the agile practices, results to enhanced relations between team members and to improved employee satisfaction metrics.

Summary. The author states that the extra challenge large Agile projects face in addressing the increased complexity of distributing and scaling development effort requires careful evaluation of organizational factors and Agile practices in order for the Agile methodologies to remain beneficial. The author states that when trying to scale Agile projects, four organizational factors are important: (a) organizational design, (b) decision making, (c) collaboration and coordination, and (d) Agile culture.

Using a case study approach of two project teams of similar large size and distribution footprint, one using Agile practices and the other using traditional development practices, the author found that adopting the Agile framework provided a greater level of quality, allowed for easier incorporation of requirements changes throughout the project lifecycle, and improved employee satisfaction while building the product as compared to the project using traditional devleopment methodologies. The author notes that adopting the Agile framework is not

straightforward for large organizations, especially those with long histories of traditional processes, and that careful planning is required to avoid common problems.

The author suggests that Agile practices used on small, collocated teams need to be reevaluated when used on large, distributed efforts. As opposed to maintining a single backlog of activities across the entire project, using multiple team backlogs that are smaller and include only tasks related to the specfic team helps teams focus on the actual working package per iteration. As the number of participants in a project grows, it becomes more difficult to hold meetings with everyone. The author recommends that teams conduct sprint planning, daily standups, and retrospective meetings individually per team, with responsibility falling on the Scrum Master to summarize feedback and bring it back to the larger project level. The author states that the infrastructure and tools such as integrated development environments, information sharing tools, and project management tools must be evalatued based on the size of the project, the number of teams, corresponding needs, and customer requirements. Lastly, the author argues that while Agile might be a software devleopment methodology, organizational agility and a mindset shift are required to support large scale Agile deployments. The author writes that “communicaiton paths, human resource policies, and management approaches must be reevaluated to better align with the team-based nature of Agile development” (p. 458).

This article is relevent to this study as it provides empirical eveidence that the adoption of Agile methods on large, distributed efforts can be successful and yield more positive results compared to traditional development methods. Several best practices and other factors to consider when adopting large-scale Agile are presented.

Turetken, O., Stojanov, I., & Trienekens, J. J. M. (2016). Assessing the adoption level of scaled agile development: A maturity model for scaled Agile framework. *Journal of Software: Evolution and Process, 29*(6), e1796. https://doi.org/10.1002/smr.1796

Abstract. Although the agile software development approaches have gained wide acceptance in practice, the concerns regarding the scalability and integration of agile practices in traditional large-scale system development projects are prevailing. Scaled Agile Framework (SAFe) has emerged as a solution to address some of these concerns. Despite few encouraging results, case studies indicate several challenges of SAFe adoption. Currently, there is a lack of a well-structured gradual approach for establishing SAFe. Before and during SAFe adoption, organizations can benefit greatly from a uniform model for assessing the current progress and establishing a roadmap for the initiative. To address this need, we developed a maturity model that provides guidance for software developing organizations in defining a roadmap for adopting SAFe. The model can also be used to assess the level of SAFe adoption. We took an existing agile maturity model as a basis for agile practices and extended it with practices that are key to SAFe. The model was developed and refined with industry experts using the Delphi technique. A case study was conducted in a large organization where we evaluated the model by applying it to assess the level of SAFe adoption.

Summary. As the authors note, organizations struggle with the full adoption of Agile development practices over a short period of time. The authors assert that maturity models can help guide organizations in providing direction on Agile practices and the manner used to introduce and establish these practices in the organization. The authors define a maturity model as a conceptual framework that outlines the stages of maturation paths and is made up of best practices that help organizations to improve their processes in a particular area. Maturity models

are characterized by an ordered number of maturity levels, with each level defining the characteristics or practices that must be achieved.

The authors of this study note that there is a lack of a structured roadmap that guides organizations on the necessary preparation and adoption of SAFe, one of the more popular commercial models for scaling Agile development across the enterprise. The research question they seek to address is: *How to design a maturity model that can be used as a guideline by software developing organizations to adopt SAFe and assess the success level of SAFe adoption*? To address that question, the authors undertook a design science research approach by developing a new software engineering artifact, the SAFe Maturity Model (SAFe MM), whose goal is to assess the level of SAFe adoption and help in defining a roadmap for the implementation of Agile and SAFe practices in the enterprise. To observe the validity of the SAFe MM, the researchers evaluated the artifact in real life settings by applying it to a large international corporation transitioning to a scaled Agile way of working. Through observed results in the assessment and post-interviews with people involved, the authors confirmed the SAFe MM's ability to reveal and pinpoint the company's strong and weak points in achieving Agile and SAFe practices, along with those aspects that require immediate attention.

This article is relevant to this study as it introduces the concept of a SAFe maturity model as a best practice for software development organizations which have already begun or are considering adopting SAFe as their way of working. As the authors note, it provides large software development organizations a continuous and evolutionary approach to improve its capability in adopting Agile and SAFe practices.

van Manen, H., & van Vliet, H. (2014). Organization-wide Agile expansion requires an organization-wide Agile mindset. In Jedlitschka, A., Kuvaja, P., Kuhrmann, M.,

Männistö, T., Münch, J., & Raatikainen, M. (eds.). *Product-Focused Software Process Improvement* (pp. 48–62). Cham, Switzerland: Springer. https://doi.org/10.1007/978-3-319-13835-0_4

Abstract. While agile methods are widely used, large organizations still struggle with the implementation thereof throughout the whole organization. The objective of our study is to identify factors that affect the expansion of agile software development in large organizations. We performed a multiple-case study to do so. We found agile software development in large organizations is more than implementing Scrum. In particular, we identified "Agile mindset" as a crucial topic that deserves attention when expanding agile methods in large organizations.

Summary. Using a multiple-case study approach, the authors seek to identify factors that affect the expansion of Agile development in large organizations. Through the process of interviewing members of two large organizations undergoing transitions to Agile ways of working, the research identified that being a truly Agile organization requires more than the implementation of a single Agile method such as Scrum. According to the authors, *Being Agile* is a mindset based on collaboration, trust, and continuous improvement.

Collaboration is positively influenced by having teams fully dedicated to one development stream, coupled with experience working together so that members know each other's strengths and weaknesses. Factors negatively impacting collaboration include having multiple competing partner companies in the development process, a serialized work process, and individual thinking where a team is too focused on itself and not on the other teams they are working with to complete a product.

Factors positively impacting trust include using dedicated teams with experience working together as one team, self-steering teams, a culture of feedback and transparency, and a culture of

taking responsibility. Measuring, controlling, and accounting for the output of teams; producing lots of reports; and following extensive organizational processes all negatively affect the feeling of trust in an organization. A culture of feedback, leadership who champion Agile practices, measuring added business value over costs, and a willingness to try new ways of working are all factors that positively influence the element of continuous improvement.

This article is relevant to this study as it explores the concept of successfully expanding Agile development methods in large organizations through the adoption of an Agile mindset. Key elements and best practices for embracing an Agile mindset are included.

Conclusion

As compared to traditional, plan-based software development methodologies, Agile methodologies provide several benefits such as adaptability to changing requirements, shorter development cycles, and increased focus on customer needs (Abbas et al., 2008; Alqudah & Razali, 2016; Barlow et al., 2011; Dikert et al., 2016; Gandomani & Nafchi, 2016; Hobbs & Petit, 2017; van Waardenburg & van Vliet, 2013). While originally designed for small team projects with few interdependencies, the potential benefits of Agile software development methods have shown promise for large projects and organizations as well (Dikert et al., 2016; Paasivaara & Lassenius, 2016; van Waardenburg & van Vliet, 2013). However, challenges and concerns around the scalability and integration of these practices for large projects and organizations exist (Turetken et al., 2016; Paasivaara & Lassenius, 2016; Papadopolous, 2015). Despite concerns about the increased complexity and coordination required and the resulting impacts of organizational change, there is an increased trend of adopting Agile software development methods at scale in large projects and organizations in the hopes of reaping the benefits enjoyed by organizations employing Agile on a smaller scale (Dikert et al., 2016; Paasivaara & Lassenius, 2016). As the popularity of Agile adoption increases, large organizations must focus on the best practices of how to adopt and scale these methods (Dikert et al., 2016; Paasivaara & Lassenius, 2016).

Background and History of Agile Methodologies

In the mid-1990's most software development projects followed a heavyweight and linear development methodology consisting of a completed set of requirements and design, followed by coding and testing activities based on a thorough plan (Williams, 2012). The guiding philosophy of these projects was "Do it right the first time" (Williams, 2012, p. 71). At the time, the

common belief among software development practitioners was that as long as projects adhered to a strict methodology, they would be successful in meeting the expectations and needs of their customers (Williams, 2012). According to Williams (2012), the reality often failed to meet these expectations.

At the same time, an alternate sentiment began to gain attention, one that embraced increasingly iterative, lightweight software development methodologies such as Extreme Programming, Dynamic Systems Development Method (DSDM), Crystal, and Scrum (Williams, 2012). According to Abbas et al., (2008), these methods emerged as a reaction to the increased dissatisfaction of software developers with traditional, plan-based software development methods. In February 2001, 17 representatives and practitioners of software development and others who were sympathetic to the need for an alternative to documentation-driven, heavyweight development processes gathered at a ski resort in Utah in an effort to find common ground (Beck et al., 2001). The *Manifesto for Agile Software Development,* commonly referred to as the *Agile Manifesto* emerged, with four core values and 12 principles, and the term "Agile software development" was established (Williams, 2012, p. 72). According to Beck et al. (2001), the authors of this *Agile Manifesto* sought to uncover better ways of developing software; through their work they came to value the following:

- *Individuals and interactions* over processes and tools.
- *Working software* over comprehensive documentation.
- *Customer collaboration* over contract negotiation.
- *Responding to change* over following a plan (para. 2).

Abbas et al. (2008) argue that while Agile software development has been most visible since the introduction of the *Agile Manifesto* in 2001, 'Agile' ideas, practices, and experiences

have existed for decades longer. From the National Aeronautics and Space Administration's (NASA's) use of iterative and incremental development in the 1960s; Winston Royce's recommendations in 1970 of regular customer engagement and the use of pilot projects; to James Martin's Rapid Application Development (RAD) methodology and its fundamentals of iterative development, small high-skilled teams, and quick delivery; Abbas et al. (2008) contend that Agile methods are more than a recent software development fad. Instead, they argue that Agile development is an evolution of approaches that have been successful in practice for many decades. Research performed by Williams (2012) indicates that this success has been enduring; even though several evolutions of Agile software development practices have occurred since its publication, the core principles found within the *Agile Manifesto* remain highly valued, supported, and practiced within the software development community.

Challenges of Large-Scale Agile Implementations

According to Hobbs and Petit (2017), larger information technology projects and organizations are faced with unique organizational challenges that come with the coordination of multiple development teams, organization of a greater number of specialists that exist outside of the development teams, and the integration that is required with existing information systems. These complexities lead to challenges when adopting Agile methods in large projects and organizations (Dikert et al., 2016; Hobbs & Petit, 2017; Turetken et al., 2016; Paasivaara & Lassenius, 2016; Papadopolous, 2015). Dikert et al. (2016) noted that one of the most prominent challenges in large-scale Agile transformations was the difficulty in coordinating the work across several Agile teams. Their research found that many challenges arose when teams needed to work with other teams and as part of the larger surrounding organization. In their research on attempts by organizations to implement Agile on a larger scale, Dikert et al. (2016) found that

while Agile methods had created flexibility at the team level, the surrounding organization was not responsive enough to enable complete adoption of Agile practices and interdependencies still existed which made managing development activities difficult. Saeeda, Minhas, and Humayun (2015) found similar challenges in implementing Agile for larger projects and organizations stemming from the need to coordinate and communicate between multiple teams, while also adding that large-scale Agile software development generally leads to an increased number of releases and therefore increased overhead and maintenance. Van Waardenburg and van Vliet (2013) found the lack of business involvement in Agile projects to be a common challenge with large enterprises. They argue that this lack of engagement occurs because large organizations often centralize the IT department, creating a gap between business and IT functions (van Waardenburg & van Vliet, 2013).

Hobbs and Petit (2017) note the implementation of Agile methods in large, well-established organizations have significant impacts on organizational change in terms of structure, process, and culture. Changing role definitions and change approval processes and a shift from command-and-control to servant leadership styles all represent challenges many organizations face when implementing large-scale Agile transformations (Hobbs & Petit, 2017).

Paasivaara and Lassenius (2016) also found that commercially available frameworks specifically designed to scale Agile methods in large organizations and projects were not always successful. In their research, they found that in addition to project complexity, a failure to adjust organizational culture and embrace Agile practices made it more difficult for large organizations to achieve desired results from adopting an Agile scaling framework.

Best Practices of Large-Scale Agile Implementations

Gandomani and Nafchi (2016) argue that the most important transition challenges are not technological in nature but rather social and people-oriented. They describe seven essential prerequisites organizations and their leaders should embrace prior to executing large-scale Agile transformations: (a) having a convincing reason for change; (b) people buy-in; (c) defining business goals; (d) initial training; (e) pilot project selection; (f) pre-startup assessment; and (g) team set up. Gandomani and Nafchi (2016) contend that effecting these prerequisites better prepares people and organizations to deal with the potential challenges resulting from Agile transformation.

In their study, Dikert et al. (2016) found the success categories most mentioned as important for an Agile transformation process include choosing and customizing the Agile approach, management support, mindset and alignment, and training and coaching. Mindset and alignment included such factors as concentrating on and embracing Agile values, aligning the organization towards a common goal of introducing new development methods, and the formation and support of Agile communities (Dikert et al., 2016). Paasivaara and Lassenius (2014) also recommend the use of communities of practice (CoPs) as a means of facilitating the scaling of software development through Agile transformations. Based on their research, they found that CoPs can be used effectively as support mechanisms, providing a forum to discuss the transformation, collaborate on methods for continous improvement, and faciliate knowledge management practices. CoPs also support the scaling of Agile methods throughout large and distributed organizations, including efficient coordination and knowledge sharing between teams (Paasivaara & Lassenius, 2014).

Papadopolous (2015) agrees that organizational agility and a cultural mindset shift are required to support large-scale Agile implementations. He writes that reevaluation of communication paths, human resource policies, and management approaches are all necessary to better align with the team-based nature of Agile development. Van Manen and van Vliet (2014) suggest that an Agile mindset is a crucial factor when expanding Agile methods in large organizations. Further, they suggest that an Agile mindset is based on collaboration, which is positively influenced by having teams fully dedicated to one development stream; trust, which is enhanced with the use of dedicated teams with experience working together; and continuous improvement, which can be facilitated by developing a culture of feedback, championing Agile practices, and measuring business value over costs (van Manen & van Vliet, 2014).

Alqudah and Razali (2016) note that different Agile scaling methods and frameworks have been developed in order to help address common issues with team quantity and size, organizational interdependencies, and other issues such as lack of customer interaction and involvement that often arise in large-scale Agile adoptions. They identify Scaled Agile Framework (SAFe), Disciplined Agile Delivery (DAD), and Large Scale Scrum (LeSS) as examples of these frameworks and note that these and other Agile scaling approaches use a combination of Agile and Lean methods to overcome common challenges in large-scale Agile adoption. For example, SAFe incorporates practices from Scrum, Extreme Programming (XP), Kanban, Lean and DevOps into a prescriptive framework that utilizes portfolio management tools and principles. This prescriptive structure provides an advantage for many organizations as it helps ease adoption and is designed to support large enterprises with multiple teams working together (Alqudah & Razali, 2016).

Turetken et al. (2016) recommend the use of a maturity model for organizations seeking to implement Agile practices in large-scale system development projects. They define a maturity model as a conceptual framework that outlines the stages of maturation paths and is made up of best practices that help organizations to improve their processes in a particular area. Turetken et al. (2016) note that maturity models are characterized by an ordered number of maturity levels, with each level defining the characteristics or practices that must be achieved.

Barlow et al. (2011) argue that both traditional plan-based methods and Agile approaches have strengths and weaknesses in software development. They assert that plan-based methods are more suitable for projects large or small where interdependencies are sequential and result from the serial nature of the workflow, while Agile methods are more effective in projects with small team sizes with reciprocal interdependencies that exist when two or more parties depend on each other in both directions. They recommend the use of hybrid methods, or those that combine aspects of both plan-based and Agile methods, in large-sized projects with many reciprocal interdependencies. Barlow et al. (2011) assert that large projects often found in larger, mature organizations will be able to leverage some of the benefits of Agile development without sacrificing the stability provided by traditional methods.

Final Thoughts

Agile software development offers numerous benefits for organizations, including accelerated time-to-market, the flexibility to respond to frequently changing requirements and priorities, better alignment with business goals, improved productivity, and increased customer satisfaction (Abbas et al., 2008; Alqudah & Razali, 2016; Barlow et al., 2011; Dikert et al., 2016; Gandomani & Nafchi, 2016; Hobbs & Petit, 2016; van Waardenburg & van Vliet, 2013). However, implementing Agile methods in large organizations presents greater challenges than

implementing in small organizations, in part because of the greater number of dependencies and coordination required between projects and teams and the often difficult changes to organizational and cultural mindsets (Turetken et al., 2016; Paasivaara & Lassenius, 2016; Papadopolous, 2015).

The literature presented in this Annotated Bibliography is intended to inform IT leaders who are considering adopting Agile practices on a large scale by placing focus on three categories: (a) background and history of Agile methodologies, (b) challenges of large-scale Agile implementations, and (c) best practices of large-scale Agile implementations. By leveraging the research contained within, IT leaders who are considering large-scale Agile transformations will be able to develop better strategies and tactics, resulting in a higher likelihood that their implementations of Agile methods will succeed.

References

Abbas, N., Gravell, A. M., & Wills, G. B. (2008). Historical roots of Agile methods: Where did "Agile thinking" come from? In *Agile Processes in Software Engineering and Extreme Programming* (pp. 94–103). Heidelberg, Berlin: Springer. https://doi.org/10.1007/978-3-540-68255-4_10

Alqudah, M., & Razali, R. (2016). A review of scaling Agile methods in large software development. *International Journal on Advanced Science, Engineering and Information Technology, 6*, 828. https://doi.org/10.18517/ijaseit.6.6.1374

Barlow, J., Giboney, J., Keith, M., Wilson, D., Schuetzler, R., Lowry, P., & Vance, A. (2011). Overview and guidance on Agile development in large organizations. *Communications of the Association for Information Systems, 29*(1). Retrieved from http://aisel.aisnet.org/cais/vol29/iss1/2

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M.,...Thomas, D. (2001). Manifesto for Agile software development. Retrieved from http://www.agilemanifesto.org/

Center for Public Issues Education. (2014). Evaluating information sources. Retrieved from http://www.piecenter.com/wp-content/uploads/2014/08/evaluateinfo.pdf

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

Charette, R. N. (2005). Why software fails [software failure]. *IEEE Spectrum, 42*(9), 42–49. https://doi.org/10.1109/MSPEC.2005.1502528

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

Gandomani, T. J., & Nafchi, M. Z. (2016). The essential prerequisites of Agile transition and adoption: A grounded theory approach. *Journal of Korean Society for Internet Information, 17*(5), 173–184. https://doi.org/10.7472/jksii.2016.17.5.173

Gregg, S. P., Scharadin, R., & Clements, P. C. (2016). The best of both worlds: Agile development meets product line engineering at Lockheed Martin. *INCOSE International Symposium, 26*(1), 951–965. https://doi.org/10.1002/j.2334-5837.2016.00204.x

Henriksen, A., & Sven Arne R., P. (2017). A qualitative case study on agile practices and project success in agile software projects. *Journal of Modern Project Management, 5*(1), 62–73. https://doi.org/10.19255/JMPM01306

Hobbs, B., & Petit, Y. (2017). Agile methods on large projects in large organizations. *Project Management Journal, 48*(3), 3-19. Retrieved from https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/pmj/2017-june-july.pdf#page=4

Iivari, J., & Iivari, N. (2011). The relationship between organizational culture and the deployment of agile methods. *Information and Software Technology, 53*(5), 509–520. https://doi.org/10.1016/j.infsof.2010.10.008

Paasivaara, M., & Lassenius, C. (2014). Communities of practice in a large distributed Agile software development organization – Case Ericsson. *Information and Software Technology, 56*(12), 1556–1577. https://doi.org/10.1016/j.infsof.2014.06.008

Paasivaara, M., & Lassenius, C. (2016). Scaling Scrum in a large globally distributed organization: A case study. In *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)* (pp. 74–83). https://doi.org/10.1109/ICGSE.2016.34

Papadopoulos, G. (2015). Moving from traditional to Agile software development methodologies also on large, distributed projects. *Procedia - Social and Behavioral Sciences, 175*, 455–463. https://doi.org/10.1016/j.sbspro.2015.01.1223

Pawel, P. (2017). Agile transformation in project organization: Issues, conditions and challenges. Paper presented at Sixth International Scientific Conference on Project Management in the Baltic Countries, University of Latvia, Riga, 27-28 April. Retrieved from http://balticpmconference.eu/sites/default/files/image-uploads/proceeding_book_26.04.2017_final.pdf

Petersen, K., & Wohlin, C. (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of Systems and Software, 82*(9), 1479–1490. https://doi.org/10.1016/j.jss.2009.03.036

Project Management Institute (2017). *A guide to the project management body of knowledge.* Newtown Square, PA: Project Management Institute, Inc.

Saeeda, H., Arif, F., Minhas, N. M., & Humayun, M. (2015). Agile scalability for large scale projects: Lessons learned. *Journal of Software, 10*(7), 893-903. https://doi.org/10.17706/jsw.10.7.893-903

Serrador, P., & Pinto, J. K. (2015). Does Agile work? — A quantitative analysis of agile project success. *International Journal of Project Management, 33*(5), 1040–1051. https://doi.org/10.1016/j.ijproman.2015.01.006

Stettina, C. J., & Hörz, J. (2015). Agile portfolio management: An empirical perspective on the practice in use. *International Journal of Project Management, 33*(1), 140–152. https://doi.org/10.1016/j.ijproman.2014.03.008

Turetken, O., Stojanov, I., & Trienekens, J. J. M. (2016). Assessing the adoption level of scaled agile development: A maturity model for scaled Agile framework. *Journal of Software: Evolution and Process, 29*(6), e1796. https://doi.org/10.1002/smr.1796

van Manen, H., & van Vliet, H. (2014). Organization-wide Agile expansion requires an organization-wide Agile mindset. In Jedlitschka, A., Kuvaja, P., Kuhrmann, M., Männistö, T., Münch, J., & Raatikainen, M. (eds.). *Product-Focused Software Process Improvement* (pp. 48–62). Cham, Switzerland: Springer. https://doi.org/10.1007/978-3-319-13835-0_4

van Waardenburg, G., & van Vliet, H. (2013). When Agile meets the enterprise. *Information and Software Technology, 55*(12), 2154–2171. https://doi.org/10.1016/j.infsof.2013.07.012

Williams, L. (2012). What Agile teams think of Agile principles. *Communications of the ACM*, *55*(4), 71–76. https://doi.org/10.1145/2133806.2133823