

Corporate Innovation and Agile Project Management

Kate Davis and Jeffrey K. Pinto

Outline

1.0 Innovation and Agile: Better Tools to Address Pressing Needs Introduction to Agile

2.0 Waterfall versus Agile

2.1 Elements of Waterfall Project Management

2.2 Advantages of Waterfall Project Management

2.3 Limitations of Waterfall

3.0 The Agile Approach

3.1 Tasks Versus Stories

3.2 Steps in Agile

3.2.1 Sprint Planning

3.2.2 Daily Scrums

3.2.3 The Development Work

3.2.4 Sprint Reviews

3.2.5 Sprint Retrospective

4.0 Innovating with Agile

4.1 Agile Project Life Cycle

4.2 Agile Variations

4.2.1 Agile Scrum Methodology

4.2.2 Agile Lean Software Development

4.3 Agile Transformation (from Waterfall to Agile): Challenges for Innovation

4.3.1 Lack of Commitment to Agile

4.3.2 Lack of Knowledge and Training

4.3.3 Organizational Structure

4.3.4 Scalability

5.0 The Advantages and Disadvantages of Agile

6.0 Guidelines from Practicing Industry Professionals

7.0 Conclusions

Word Count: 9402

Broken Down into:

137 words - Title Page

840 words - Citations Page

515 words - Appendix I: Glossary of Terms

7880 words - Chapter Body

1.0 Innovation and Agile: Better Tools to Address Pressing Needs Introduction to Agile

In general, it has been recognized that an innovation is a new idea, which may be a recombination of old ideas, a scheme that challenges the present order, a formula, or a unique approach which is perceived as new by the individuals involved (Rogers, 1983). By this definition, innovations consist of both new technical and administrative ideas: Technical innovations represent new technologies, products, and services while administrative innovations represent new procedures, policies, and organizational forms (Van de Ven, 1986). In fact, Van de Ven (1986) rejected the distinction between various forms of innovation, preferring to look at them as a combination since most innovations consist of new technical and administrative components. Fagerberg and Verspagen (2009) further offer a definition of innovation as “new combinations” of existing knowledge and resources; the distinction between invention (new ideas) and innovation (implementing these in practice); the classification of innovations into product, process and organizational innovation, and the radical nature of their social and economic impacts (making distinctions between revolutionary, radical and marginal or incremental innovations).

It is against the backdrop of significant corporate and societal demands for new goods and services (the innovation mandate) that the practical means for addressing these concerns emerged in the form of projects. Early efforts to accelerate innovation emerged from Toyota, in the form of its approaches to lean and quality. These innovations tended to focus on physical products and their rapid prototyping, development, and continuous improvement. However, as the global economy grew increasingly software driven and computer-centric, methods for innovation had to move beyond the narrower product economy and address the broader marketplace. Back in the late 1980's and early 1990's, for example, a transition started to take place within the corporate environment, as it related to innovation and new product development, particularly in software and information technology (IT) systems. Failure rates for these projects, as reported in *Chaos Reports*, were disappointingly high and had remained so for years. Among the reasons for this often-abysmal record were a critical and recurring failure to gain client acceptance and use of new systems. Thus, even when projects were managed efficiently, they failed at the final hurdle, when customers for whom they were developed, often received them with lackluster response and limited actual use.

Since the turn of the millennium, traditional highly structure approaches to project management are decreasing in effectiveness for many project types and a new project planning methodology is required. As an example, consider the fields of IT (including software) and new product development, where customer/consumer needs continually change meaning the end result cannot be visualized. To accommodate this, many organizations value flexibility and proactivity to opportunities or changes mid-stream.

Agile Project Management (Agile PM) reflects a new era and differs from traditional project management by recognizing that the old approach of carefully “planning the work and then working the plan” does not take into account the reality of many modern organizations and projects. Heedlessly following the original plan throughout would make no sense as the project progresses through the execution stage. Agile PM recognizes the importance of these evolving needs and allows for an incremental, iterative planning process—one that stays connected to clients across the project life cycle.

Agile PM offers an alternative to the traditional, “waterfall” planning process, which uses a linear, sequential life cycle model following a logical series of steps. For example, under **waterfall**, each stage in a software development process follows completion of the previous stage; first, requirements are fully gathered, then the system is designed, implemented, and tested. Finally, it is fully deployed and subject to maintenance. In such a model, each phase must be completed fully before the next phase can begin. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In the waterfall model phases do not overlap. The waterfall project development process works well when:

- Requirements are very well understood and fixed at the outset of the project.
- Product definition is stable and not subject to changes.
- Technology is understood.
- Ample resources with required expertise are available freely.
- The project is of relatively short duration.

Although originally rooted in software development, agile project delivery methodology offers applications for many of the common concerns and themes in innovation (for example, new

product development projects). As firms seek to innovate to achieve competitive advantage, first-mover advantage, and technological breakthroughs, they face an equally challenging conundrum: how to most efficiently manage projects shrouded in uncertainties? Perhaps the uncertainties are technological in nature (“Can this new idea actually work?”), socio/political (“Will the public and regulatory agencies be receptive of this new product or service”), or economic (“Can this innovation actually become profitable and self-sustaining?”). Given the sometimes enormous monetary outlays necessary to complete these innovative projects, it is understandable that corporate senior managers would seek to rationalize project development cycles as much as possible, in order to keep a firm handle on cost and delivery (hitting product launch windows). Thus, we have the curious juxtaposition of innovative new products being committed to standard, rationalized project management practices that work well for deterministic (easy to plan) ventures.

But what happens if requirements change in the middle of the project’s development? Or the customer delivers a new set of “critical” features that must be part of the final product? Or when a new technological innovation allows our team to streamline the software we are working on to make it more user-friendly? What if the initial assumptions or project scope were poorly communicated or a competitor beats us to the market with the identical product? Under these circumstances, the decision to lock into a fully articulated project scope, set in stone at the outset of the project, can lead to critical problems. With more and more projects supporting new product development initiatives that need the flexibility to change course midstream, the waterfall model can lead to a rigid process that will not deliver value to the customer, either because their needs are constantly changing or because they did a poor job of initially articulating exactly what the project was supposed to do.

Consider an example: Google Glass was launched in 2013 as a product intended to bring information, in the form of a tiny, glasses-attached screen, to the public. It was marketed as the latest in mobile smart technology and Google was confident that its expensive, but trendy product would do well, once the public had a chance to experience and ultimately, embrace it. A high price, coupled with significant aesthetic, safety, and an unclear set of functions doomed the product. “Unfortunately, the Glass failed because the creators neglected to define and validate the users and what problems it was solving for them. Instead, they assumed the product would sell

itself even without real solutions or value, that its hype would be enough to appeal to everyone” (Yoon, 2018: p. 1).

The Google Glass fiasco represents another example of the reasons why Agile project development was promoted as an effective means to realize product innovation in a manner that is both efficient and market-driven. Although first set within the confines of traditional software development, its applications were quickly broadened to other classes of project for which user acceptance (and an initial knowledge of their real needs) was paramount and where poor responses could quickly cost these firms millions in capital and goodwill. Part of the reason for this poor responsiveness to new products and services was the way traditional project management was structured. That is, the industry standard process (waterfall) to develop software and other new products was often rigid and time consuming, making it difficult for the developers to keep up with the needs of the business and/or market demand. To advance a new flexible approach to project development, a meeting of seventeen software developers took place in Utah to address the recurring problems of failing projects and the *Manifesto for Agile Software Development* was the output of this meeting (Beck, et al., 2001; Rigby et al., 2016). This Manifesto identifies the twelve principles of Agile development (Source: <http://agilemanifesto.org/principles.html>):

1. The top priority is customer satisfaction through early and continual delivery of valuable products
2. Embrace change no matter what stage of the process, as this is Agile’s competitive advantage
3. Deliver frequently – with time interval from one to four weeks
4. Stakeholders and development teams must communicate and/or collaborate daily for the duration of the project
5. The projects should be built around empowered and motivated team members. Provide a safe environment to work in and the support they need to succeed
6. In person communication is the most effective and efficient way of conveying information both to and within the development team
7. Completed, functional work is the primary measure of forward progress

8. Agile processes should yield viable and sustainable development creating an environment where developers and stakeholders are able to maintain a constant pace indefinitely for the iterative work identified
9. Agility is enhanced by a continuous effort to achieve technical excellence and superior design quality
10. Simplicity is key and identifying all the work that is not needed is essential
11. Self-organizing teams create the best requirements, designs, and architecture
12. The team reflects on how to be more effective/efficient at regular intervals throughout the project, then takes actions to adjust behavior accordingly

One major theme behind these principles is that when facing uncertainty to try something that you will believe will work, get feedback, and adjust your model as needed (Beck, et al., 2001). Agile is built on the belief that this evolutionary approach is more efficient (Dingsøyr and Dybå, 2008); that is, by taking this iterative development approach, Agile does not rely on a fixed set of up-front requirements or specifications (Serrador and Pinto, 2015).

2.0 Waterfall versus Agile

2.1 Elements of Waterfall Project Management

Waterfall project management is an approach that follows a linear and sequential process to complete projects dictating that a project complete one phase prior to the next phase beginning. Common Waterfall phases are:

1. Requirements - the analysis gate of the project, reviewing the initiative, project problem definition and corporate strategy alignment
2. Design - scope definition, i.e., what the project will and will not do. “Locking down” requirements and scope
3. Change control - changes are frowned upon and any requested follow a strict change control process where each requested modification is fully vetted by a change control board, where the impact the of time, scope and cost are scrutinized prior to a change being accepted or rejected

4. Implementation - the project is executed, and the work defined in the design phase begins. The manufacture and build start here and changes follow a rigid process
5. Verification - all output from the project is rigorously tested for conformity to expected standards and initial project scope targets
6. Maintenance - the product or project falls into a warranty period to address potential fixes or upgrades. Should these arise, the project reverts to the requirements phase and the process starts anew.

2.2 Advantages of Waterfall Project Management

The Waterfall methodology offers several strengths for project execution. First, the method follows a logical and sequential pattern. Fair (2012) noted benefits that include the linear-phased approach creating dependencies that help connect project phases clearly, thus allowing both stakeholders and project developers to have a clear view of the current phase and anticipate what is next. The project's progress (or lack thereof) can be pinpointed, diagnosed and triaged. Second, the heavy upfront planning can help cut down on unnecessary changes to scope and garners stakeholder agreement at the start of the project. Traditional project planning can be used as a catalyst to get the project moving in the right direction and to develop a strong backlog (prioritized lists of tasks or user stories that are waiting to be completed) (Quist, 2015). Third, quality is planned upfront so there that fewer "surprises" when products are delivered. Finally, Waterfall provides project transparency in the form of documentation, phase signoff, and storage of project artifacts. That is, the waterfall methodology promotes the requirement for formal documentation (Quist 2015).

2.3 Limitations of Waterfall

While the obvious appeal of waterfall lies in the linear, progressive nature of a locked-in development cycle, there are several limitations and challenges with applying waterfall. With this methodology, for example, no working product is produced until most of the work steps are completed, creating a "take it or leave it" scenario during the acceptance phase. Therefore, the quality and detail of the requirements is critical in waterfall deployments. The requirement

gathering sessions for large projects can last weeks and if the subject matter experts are not engaged, then requirements can be missed. Since there is little flexibility in the process, there is a tendency for teams to add more features than those needed as they know this is their chance to get what they want (Boehm, 1996). This in a sense, is a variation of scope creep, and can lead to costly “nice to have” features and functionality, which can add complexity to the design and increase delivery time (Pinto and Serrador, 2015). Change requests must follow the same approval process as the business case and if submitted after the design has been completed, are often not worked on until after the original business case is completed.

Other problems with planning and delivery include rigidity and lack of flexibility of the mindset extinguishing creativity and limiting stakeholder input or making stakeholders feel unheard as focus is on systems rather than people (Nichols et al., 2015; Quist 2015). Stakeholder input is also collected at the beginning of the project and the end user generally gets the “finished product” as opposed to being able to comment incrementally. The research by Nichols et al. (2015) studied the appropriateness of choosing a project methodology for small teams in both academia and industry. They found several shortcomings with the Waterfall method. The study cited that all the project features (from high to low priority) are designed and developed before any are implemented, which limits stakeholder input.

Voluminous documentation requirements and process recording can also slow project progress and place a drain on resources and a significant investment has been made in the project before results are ever produced or tested. A failure anywhere in the project life cycle may jeopardize the scheduled completion.

3.0 The Agile Approach

One of the critical differences between Agile and traditional project planning relates to the nature of the roles that key members assume. In a traditional project planning process, the project developer’s viewpoint is considered most important. Developers are looking at the project from the inside perspective; that is, how long will development take? How many work packages and tasks are necessary to complete the project? Project developers are interested in tasks because they allow them to accurately and efficiently plan the work and build their cost estimates. The more

detailed and specific the project is made, the easier it is to create these plans and estimates.

3.1 Tasks Versus Stories

User stories are different and very important for understanding the real needs of the client (“voice of the customer”). They are written by or for customers as a means for influencing the development of the functionality of the product. The more the customer can explain what they do, what they need, and how they can make use of the product to do their job better, the clearer the user story becomes and the better able the Scrum team will be to achieve these goals. Stories have value because they identify the actual work that needs to be done with the completed product or system. Once stories have been validated, they can then be decomposed into tasks. The key lies in recognizing the need to first listen to user stories to identify the specific value-added the project will provide. For example, a project that ignores the voice of the customer can be well-managed, brought in on time and under budget, but provide no real value to the customer because it was done without regard to first identifying critical user stories (what they need to do their job better).

Another Agile characteristic that demonstrates the importance of the voice of the customer is the emphasis on product **features**, as opposed to creating a detailed product Work Breakdown Structures. We assume in an Agile environment that project scope characteristics are inevitably going to change over the course of the project’s development. As a result, it makes little sense to invest heavily in time and effort to develop a sophisticated statement of scope for the overall product. Instead, Agile focuses on getting the product features right: the pieces of the product that deliver functional value to the customer. Listening to customers’ stories and defining the project’s critical deliverables in terms of features reinforces the critical nature of the ongoing, closely linked relationship the Scrum team must maintain with clients.

3.2 Steps in Agile

The Agile process follows a series of steps that allow the methodology to combine the flexibility needed to respond to customer needs with a formal process that creates a logical sequence in employing Agile planning (Schwaber and Beedle, 2002). The Scrum process involves a set of meetings that manage the project development process through (1) Sprint Planning, (2) Daily

Scrums, (3) Development Work, (4) Sprint Review, and (5) Sprint Retrospective. During the Sprint, three guidelines shape the process:

1. No changes are made that would endanger or modify the Sprint goal. Once goals for the Sprint are agreed to, they are not to be altered in the middle of the Sprint
2. Quality goals do not decrease. During the Sprint, the team cannot modify the goals or sacrifice quality standards that were first agreed to
3. Scope may be clarified and renegotiated between the product owner and the development team as more is learned. As the team discovers technical problems or opportunities during the Sprint, they are passed to the product owner for consideration on whether or not to modify the project scope.

3.2.1 Sprint Planning

The Sprint Planning session identifies the work to be performed through collaborative work of the entire Scrum team. Sprint Planning is time-boxed to a maximum of one full day (eight hours) for a one-month Sprint. For shorter Sprints, less time is required for Sprint Planning. The Scrum Master ensures that the event takes place and that all members of the Scrum team understand its purpose. The session introduces the Sprint Backlog to the development team and coordinates their efforts to achieve the Backlog items. Sprint Planning answers the following questions:

- What can be delivered in the increment (time-box) resulting from the upcoming Sprint?
- How will the work needed to deliver the increment be achieved?

3.2.2 Daily Scrums

The Daily Scrum is a short (15 minutes) event that allows the development team an opportunity to synchronize their activities and create a plan for the next 24-hour time window. During the meeting, members of the development team explain what they accomplished in the past 24 hours to meet the Sprint goal, what they intend to work on during the current day and identify any problems that might prevent the development team from completing the next Sprint goal. Daily Scrum sessions are for information purposes; they only intend to keep team members in the communications loop and identify any positive or negative trends affecting the project. Daily

Scrums also include reference to the Burndown Chart, detailing the latest information on the status of Backlog items completed (“burned down”) since the last Scrum meeting.

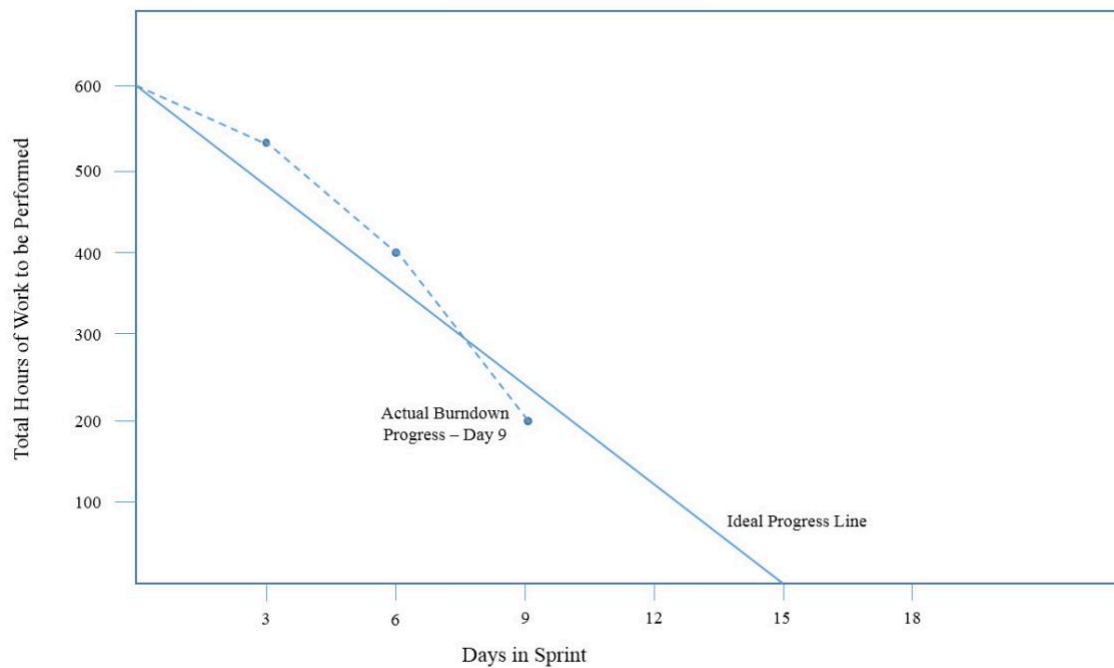
3.2.3 The Development Work

The Development Work is the time when the actual work of the project is done during the Sprint. These are the set of goals that must be achieved during the Sprint and are represented on the Burndown Chart as either in progress or completed. Development Work must be heavily coordinated between Scrum team members to ensure that no efforts are being wasted or work is being done on non-Sprint items. Figure 1 shows an example of a Burndown Chart for a Sprint that assumes the following details:

- Sprint duration – 15 days
- Team size – 5 members
- Hours/day – 8
- Total capacity – 600 hours

Figure 1: Sample Burndown Chart

Source: Pinto, J.K. (2019) *Project Management: Achieving Competitive Advantage*, 5th Ed. Pearson: Upper Saddle River, NJ. Page 394.



3.2.4 Sprint Reviews

A Sprint Review is held at the end of the Sprint to inspect the completed increment (the Sprint Backlog) and make changes to the Product Backlog if needed. During the Sprint Review, the Scrum team and other key stakeholders work closely to verify what was done on the Sprint. Based on these results and any subsequent changes to the Product Backlog, the Scrum team now plans the next things to be done to add value, including product features that need completing or modifying. Sprint Reviews are informal meetings; they are not status meetings, and the presentation of the completed Sprint Backlog is only to encourage team feedback and collaboration. The result of the Sprint Review meeting is a modified Product Backlog, and an idea of the Sprint Backlog items that will be addressed in the next Sprint. During the Sprint Review, the activities to be addressed include the following:

- The product owner explains complete and incomplete Product Backlog items
- The development team discusses what went well during the Sprint, problems it ran into, and how those problems were solved
- The development team demonstrates complete work and answers questions about the latest

Sprint

- The product owner discusses the current Product Backlog status and predicts likely completion dates based on progress to date (if needed)
- The entire group collaborates on what to do next, so that the Sprint Review provides useful input to subsequent Sprint Planning
- Review how the marketplace or potential use of the product might have changed the next most valuable thing to complete
- Review the timeline, budget, potential capabilities, and marketplace for the next anticipated release of the product.

3.2.5 Sprint Retrospective

The Sprint Retrospective meeting is held to evaluate how the previous Sprint went; what worked, what did not work, and where potential improvements can be made to the Sprint process. A valuable Sprint Retrospective should also include an action plan for identifying and implementing improvements to the process. The Scrum Master works with the Scrum team to constantly improve communications and identify improvements that will be introduced during the next Sprint. In this way, each Sprint is not only about getting the work (the Product Backlog) done, but also used to generate enthusiasm for the next Sprint while creating a more efficient and productive team.

4.0 Innovating with Agile

Although there are multiple variations of Agile, the basic building blocks of Agile remain consistent. With Agile, the working team is cross-functional and are made up of members from different parts of the organization that come together to work on a specific goal. This team should include a representative for the end-users and/or customers. Agile team members are not assigned specific roles but are supposed to be self-organizing. The team works collaboratively and seek to deliver solutions in an iterative and adaptive manner (Beck, et al., 2001). To foster this collaboration, it is often recommended that the team work in the same location and has daily meetings to discuss goal progress and any roadblocks or delays. Work is completed

incrementally and in short time frames (frequently referred to as sprints or iterations). At the end of the sprint or iteration, feedback is gathered from stakeholders, and the team would fold the feedback into the next iteration. The goal is to have a functioning viable product at the end of each sprint that evolves and improves through each iteration until its final state.

4.1 Agile Project Life Cycle

The most basic and common stages of agile methodology are plan, design/develop, test, deliver and review. Because the work is broken down into smaller increments, each of these stages are much shorter in duration than traditional methodologies. These agile stages are not starting and stopping points, as agile is grounded in continuous improvement, therefore after the last stage is completed, the team immediately starts to plan for the next iteration.

4.2 Agile Variations

There are differences in the framework among the variations of agile methodology. These frameworks are grounded in the agile philosophy, values and principles however the way work is executed and managed is slightly different. These methodologies include Crystal, Dynamic Systems Development Methodology, Feature Driven Development, Extreme Programming, Kanban, Lean Software Development and Scaled Agile Framework (SAFe), (Chow and Cao, 2008). No variation is arguably better than the other, but some are better suited for a particular working environment than others. For example, Lean Software Development and Extreme Programming are designed for software deployments, however SCRUM and SAFe have been consistently leveraged for projects outside of the software development space and are commonly applied to new product development and other service innovations. Below is an overview of a few of the most popular versions in use today.

4.2.1 Agile Scrum Methodology

Although estimates vary, agile scrum methodology is practiced more often than other methodologies in firms that have implemented Agile (Dikert, et al, 2016). Scrum methodology

gets its name from Rugby where the team rallies around the ball to carry it forward and places priority on collaboration and teamwork. Scrum is believed to be first used in relation to software development (Takeuchi and Nonaka, 1986). The scrum framework was created to provide an incremental and iterative approach to manage complex work, leveraging frequent assessment and adaption in an empirical way to drive the work (Takeuchi and Nonaka, 1986). This model places importance on the customers/stakeholders and ensuring that the work is being driven by them, while acknowledging that their requirements and initial conditions are highly likely to change. There are four main roles in scrum which are the stakeholders/customers, product owner, scrum master and development team (Chow and Cao, 2008). Scrum places importance on gaining continuous feedback from the customer/stakeholder to ensure alignment. The customer/stakeholders work with the product owner who acts as their representative to the scrum team. The scrum team is facilitated by the scrum master who keeps the team on task and functioning within the agreed time frames and framework, (Chow and Cao, 2008). In scrum the planned or scheduled iterations (also known as time-box) are referred to as “sprints” (Chow and Cao, 2008) and are detailed in section 3.2.

4.2.2 Agile Lean Software Development

The term, “lean software development” was coined in 2003, by Mary and Tom Poppendieck in their book, *Lean Software Development: An Agile Toolkit*, which highlighted lean principles and tools and how they complement agile practices. Both lean manufacturing and agile methodologies believe in iterative and continuous improvement, with heavy emphases on eliminating waste to improve value to the customer/stakeholders (Poppendieck and Poppendieck, 2003). Lean software development provides a streamlined approach, by eliminating unnecessary waste and activity which in turn, allows for more product to be delivered in less time. However, achieving this requires a dedicated/specialized team which makes it harder to scale (Baseer, et al., 2015). The five phases of Lean Software Development are Analysis, Design, Implementation, Testing, and Evaluation (Poppendieck and Poppendieck, 2003). The seven Lean Software Development principles are summarized below (Poppendieck and Poppendieck, 2003):

- Eliminate Waste - Reduce defects and unclear requirements, minimize unnecessary or scrap code, avoid starting work that is not completed and task switching

- Build Quality in Incremental Development paired with Continuous Feedback - Leverage test driven development and automate manual tasks
- Create Knowledge - Provide the tools and infrastructure to document and catalog valuable learnings, code reviews, knowledge share and training
- Defer Commitment - Do not commit to decisions months in advance or without fully understanding the requirements of the business. Continue to collect and analyze data
- Deliver Fast - Avoid unnecessarily complex business requirements or solutions that are over engineered
- Respect for People Empower teams, promote productive conflict, communicate effectively and operate transparently
- Optimize the Whole - To deliver value quickly to customers, the value stream (end to end process flow) needs to be optimized, but before it can be optimized it must be understood.

4.3 Agile Transformation (from Waterfall to Agile): Challenges for Innovation

A transition from waterfall methodologies to agile can be a complicated undertaking and the larger the corporation and typically the more diverse the work is that they manage, the larger the lift is. As more and more firms recognize that integrating technological advancements in the workplace requires greater flexibility than traditional methods can provide, as well as the ability to deliver product more rapidly to keep pace with evolving customer needs and market demands (Almeida, 2017). The transition from traditional waterfall methods to Agile is not always easy, as Agile is a philosophy and therefore requires a change to not only the way we work, but how we think about work. As Laanti, et al. (2011) noted, Agile transformation is an organizational culture change. Any organizational change can face challenges, but the transition from waterfall to agile should not be simply considered a process change. It is also a change that impacts more than just the teams that are supporting the Agile work. Middle managers, stakeholders within the business, as well as upper leadership all need to buy in and support the change and the new ways of working.

Often when firms wish to be more innovative and adaptive, they rush to implement agile, so that they can reap the purported benefits of shorter delivery times (Doshi and McGregor, 2018; Pinto and Serrador, 2015). When organizations implement change, there will always be some level of resistance. Changing to agile is no different in that regard. According to research, some of the top challenges are listed in the following subsections (Dikert, et al, 2016).

4.3.1 Lack of Commitment to Agile

An agile transformation requires leadership to understand what agile is. The way the transition is implemented can impact how well it is adopted by the organization. Agile is supposed to grow “organically” within a corporation, not forced down as a mandate (Beck, et al., 2001). When teams are given a direction without leaders defining a clear path or plan to implement, then some team members may not believe the firm is truly committed to agile and that the direction is more of a whim (flavor of the month). Others will become anxious, agitated, and skeptical over the lack of clarity and start to push back (Dikert, et al, 2016). Leadership does not always understand what is required of them to support the new way of working. Challenges arise when leaders hold teams to prior delivery plans made before the conversion, requiring firm commitments, and not balancing the workload for the new agile team (Dikert, et al, 2016). This is an issue during the transformational period, as teams that are overloaded with work and/or struggling to meet aggressive deadlines, cannot learn a new way of working (Dikert, et al, 2016). In bottom-up agile deployments, challenges emerge when the management layer of the organization has not been included in the transformation planning and therefore does not feel vested in its success (Dikert, et al, 2016). In this case, the teams will struggle to get the support they needed to push work forward and work will stall.

4.3.2 Lack of Knowledge and Training

A clear priority when converting to a new methodology is to train all the teams involved – not only the working teams, but the teams that would be expected to support these teams. However, when firms are short staffed and/or ramping up, training may be deprioritized over work. Training is also a cost and is often postponed due to budgetary constraints (Miller and Haddad,

2012). Even when workers are trained, there could be challenges faced when applying the concepts learned. Challenges arise when there is a lack of coaching or guidance from subject matter experts for teams to help them apply the learned concepts to the actual work they were managing (Dikert, et al, 2016). These challenges can result in team members not understanding the roles of the team or their individual and collective responsibility for the work to be done, resulting in poor productivity. This is especially true in firms where team members are accustomed to functioning in a prescribed way. For example, when we consider the role of a developer, where, in a traditional role they would sit within the IT team and have little exposure to other teams in the organization, they may not have the familiarity of larger organizations functions and/or business acumen of their working peers and feel lost (Almeida, 2017). Lack of training for the management teams that support or interact with the working teams is another challenge. Often this is not seen as a priority and treated as a “lunch and learn” type of training which may not give the level of understanding needed to support the new way of working.

4.3.3 Organizational Structure

We noted previously that the transformation to agile affects all work groups within the firm. Therefore, firms that operate in more of a hierarchical manner will struggle with this change as they are not structurally capable of supporting an agile framework. Silos are the bi-product of these traditional, hierarchical organizations, where work is assigned from the top down with little cross pollination of ideas or knowledge sharing. These siloed teams often compete against each other for recognition and bragging rights for new innovations. This makes collaboration difficult and cooperation is only obtained by escalating upward. In fact, Doshi and McGregor (2018: para. 29) stated, “The stronger an organization’s silos, the more people will solve for the needs of their silo, versus the needs of their team.” These work habits come into direct conflict with the new ways of working as knowledge sharing, collaboration, and cross-functional teams are paramount to Agile’s success (Almeida, 2017). The hierarchical structure often does not empower teams to make decisions or resolve conflict, so these skills may be lacking even at the mid-tier levels of management. Role ambiguity also becomes a pain point during agile transformations as in traditional organizational structures roles and responsibilities are well-defined. It is no wonder

that there is confusion, conflict, and frustration when these team members are now placed in cross-function teams and expected to work autonomously.

4.3.4 Scalability

Managing one agile team may not cause a ripple in a larger organization, however challenges often arise when organizations are faced with supporting multiple agile teams. Part of the issue is that agile teams are set up to work autonomously but are dependent on other teams for some work items. Thus, when multiple agile teams are requiring work items from these same teams a bottleneck is created. To further complicate the issue, these external teams have their own work and their own set of priorities. As noted by Dikert, et al. (2016), there is a difficulty in “striking a balance between self-organizing agile teams focusing on their own goals and those of the broader organization. Lack of respect for the wider organizational context can result in coordination challenges” (quoted in Tendedez, et al., 2018, p.171-16). Teams are thus placed in the position of competing for the same resources. During a transformation when not everyone is as familiar with agile methodologies, they may not see the incremental work from the agile teams as important as the “essential” or “foundational” work it is competing with. From a software development perspective, agile practices can lead to downstream system issues, developers writing code supporting agile work can conflict with teams who are trying to create a more unified architecture (Dikert, et al, 2016). Testing can also be a challenge when scaling agile. As part of the iterative process, agile teams must have access to a testing environment. Because this is not always taken into consideration, it can become another source of delays.

5.0 The Advantages and Disadvantages of Agile

Agile strategies provide superior customer value in a changing environment (Readman et al., 2006), partly due to a hyper-collaborative stakeholder interaction philosophy. Because it does not require the same documentation and process rigors of Waterfall, data suggests that Agile products are developed and get to market at a much faster rate, with serious business implications and can be used a competitive advantage (Cooper and Sommer, 2016a). The advantages of Agile PM and particularly the Scrum-based approach is its simplicity (Cervone,

2011). Agile self-managing teams are advantageous as they promote more satisfied employees, with lower turnover, and lower absenteeism (Nils et al., 2010). The level of stakeholder engagement is consistently cited as a satisfier and contributor to project success (Kisielnick and Misiak, 2017) with flexibility allowing clients to provide feedback to the team to change their minds, add, remove, and change features without strict change control. There is also a large degree of transparency to the project schedule as the project sprints are fixed within a certain time-box. At a higher level, research from Hobbs and Petit (2017) revealed that the real advantage of using Agile approaches identified in the literature included a working environment that supports team creativity and productivity, rapid adaptation to change, and value for the customer because of better identification of needs.

Despite the recognized benefits of agile, there are some shortcomings that need to be considered, particularly as they relate to corporate innovation policies. Research from Nils et al. (2010) indicated that self-autonomous teams can be difficult to implement, and they risk failure when used in inappropriate situations or without enough leadership and support. Another challenge within agile teams is the need for constant communication. As noted by the research of Levesque et al. (2001), when the roles that group members play become increasingly specialized, there is a corresponding decline in the amount of time that team members spend working or communicating with each other. Not all individuals, team and management support, have the skill set to be able to accept the rapid fire (negative and positive) feedback and as such, may withdraw from the team. Agile uses a top-down approach in defining and estimating scope to eliminate the waste (time and money) of spending many weeks defining requirement details that may never be implemented or that may be outdated by the time the team is ready to build the feature. However, at some point in the cycle, scope needs to be locked down. If it is not, the cycle will continuously loop, and the next phase will never be reached (Sliger 2010). It is also difficult for novice agile project managers to enforce some of the most important facets of Agile, such as not requesting or making changes during a sprint (Cervone, 2011). While client feedback is one of the central tenants of Agile, it needs to be managed so that it does not become overburdening. A Rally Software Development study found that the most common question asked by project managers, upon hearing of this new agile approach to planning and scope management, is “What am I supposed to tell my boss/customer when he asks when we’re going to be finished?” (Sliger, 2006, p.7). Agile PM’s lack of documentation may also pose a problem

as audits (both internal and external) generally require a full review of project documentation. This can be overcome by partnering with the auditors ahead of time to ensure that project documentation is created and retained. Finally, there are cultural hurdles to successfully implement Agile. The “Agile Mindset” must be adopted at top management level and a servant leadership style enacted where the leader places the focus of the team first, understands their challenges and works diligently to remove impediments that prevent the team from performing at the highest level.

Agile offers some interesting commonalities and implications for another standard delivery system for new product development projects and other forms of organizational innovation: the stage-gate model popularized by Cooper (1990). Stage-gate, in its original conceptualization, is a top-down, macro model that shares many characteristics with waterfall project management: most notably, a structured process model that incorporates a series of decision gates, which follow steps in new product development, from the initial idea, through assessment, business case development, development, testing and validation, to final product launch. Cooper argued that the lack of uniformity (discipline) in the new product development process doomed many new ventures at the outset. On the other hand, stage-gate can be molded to incorporate Agile processes as well. For example, Cooper and Sommer (2016b) have argued that stage-gate and Agile offer a hierarchical organizing method that works symbiotically, combining the advantages of stage-gate macro organization (across projects and nested families of new products) with the flexibility of immediate user responsiveness and ability to quickly pivot to embrace new goals and requirements. As Cooper and Sommer (2016b: p. 521) noted:

“When Agile and Stage-Gate are integrated, and then utilized for manufactured or physical product development (hardware), Agile is normally applied as the project management method within the stages in Stage-Gate. That is, the stages remain, and Agile is applied within some stages. The Scrum method is the particular Agile method that seems most appropriate for hardware development, and indeed, is the method used for all the hardware case studies uncovered so far in industry.”

Thus, it is clear that Agile is not intended to replace a stage-gate approach to innovation so much as it is a complementary project management methodology that can offer the best of both worlds:

a method for managing a structure across stages as well as the agility to adapt to new requirements and user needs *within* the stages.

Another interesting area for further development relates to the scalability of Agile solutions to innovation projects, not as a function of number of agile teams, but in a broader sense. That is, we previously noted that Agile originated within software engineering and has a strong track record for incremental, more-specific or limited projects (what Hobbs and Petit (2017) refer to as the “Agile sweet spot”). Indeed, Hobbs and Petit (2017) noted that the transition of Agile into larger organizations, or to its use on a more macro, organization-wide level, is fraught with risks, many of which are related to recognizing that the transition to Agile is a radical change that puts extra pressure on business unit managers to prioritize their needs, commit to an innovation mindset, assign sufficient resources and autonomy to their Agile teams. This sense of ambiguity regarding the efficacy of Agile in large settings or with an assortment of product groups (not just software, but services and other physical products) pose some key research questions ripe for future study.

A final question for future research links the firm’s innovative behavior to Agile practices. Research currently suggests that project teams with positive internal climate, including participative safety, support for innovation, and task orientation are significantly correlated with R&D project performance (Pirola-Merlo, 2011). The conditions that characterize Agile project teams and techniques can be nurtured when attention is paid to the manner in which the teams are organized and supported. These Agile teams, in turn, are often successful at completing R&D or other innovation projects. It would be useful to explore additional research that directly applied agile methodologies to a range of project ventures; for example, projects whose uncertainty level assigns them differential placement on a continuum from non-innovative to highly complex and uncertain. Assessing the viability of Agile practices across this range of innovation levels would yield important knowledge for how to get the most out of agile in project innovation settings.

6.0 Guidelines from Practicing Industry Professionals

Recognizing that challenges relating to Agile include the process of transitioning from traditional, waterfall methods to this approach, we undertook a brief set of interviews with industry professionals to ascertain their thoughts on how best to manage the process, as well as soliciting any advice and/or guidelines they could offer to those considering adoption of Agile. The following are a representative set of quotes from some of these interviews:

Technical and Operations Manager

“If I have to establish guidelines for agile implementation, I would identify the client profile before the kickoff, legal issues impacts in methodology, and I like to combine some of the characteristics of the methods varying some of them, such as SAP to show how it would be when finished in the kickoff, or in the SCRUM method, varying the time samples of sprints or daily meetings, who said it is obligated to be a certain number of weeks in huge projects, or the daily meetings should be daily, instead of two times a week for instance, and the concept of vision of scope, cannot be used in some approaches to avoid future problems.

There are basically two guidelines: Instruction of new employees and how the project workflow is set up. This should be a detailed guideline to ensure every team member is aware of the entire workflow. The second guideline which we set up are standardized agenda items in meetings to make sure relevant information is addressed and non-relevant is moved to other communication channels.”

7.0 Conclusions

Projects continue to drive corporate innovation and serve as the principle engine for delivering value-added to modern organization, both public and private. Because innovation (technical and administrative intermingling, in the eyes of noted scholars such as Van de Ven, (1986)) remains a sine qua non for organizational competitiveness, it is paramount that promising new methodologies are investigated to demonstrate how they can be adapted to spur and support future organizational innovation. And yet, there is an underlying paradox in the manner by which projects have been historically managed; i.e., project failures rates remain depressingly high across most classes of project. In 2017, PMI research indicated that 28% of projects were deemed failures and only 60% of projects met their stated goals. Thus, organizations with a clear

mandate to emphasize innovative product and service solutions are both supported and to some sense, constrained by the project management efforts in support of their ventures.

It is due to an emphasis on projects and a concomitant recognition of their limitations that Agile has been found to support new innovations in organizations. Firms are looking for ways to maintain a competitive advantage and many executives believe that adopting Agile practices, they can deliver products faster and at a high quality. The 2017 PMI study also indicated that 71% of the firms surveyed were using Agile approaches for their projects sometimes, often, or always (PMI 2017). Research performed across industry does seem to provide evidence that Agile methodologies can result in greater project success within certain industries – particularly IT or those with frequently changing requirements or the desire for early working prototypes. When goals and solutions are unclear and there is high volatility, there is a particular need for alternative approaches to managing projects (Fernandez and Fernandez, 2008). Despite the evidence that Agile projects have higher success rates than Waterfall, Agile adoption does not appear to be a “silver bullet remedy” for project failure. As the research of Serrador and Pinto (2015, p.1050) point out, “(Agile) has achieved best success to date within certain settings; notably, high technology, healthcare, and professional service.”

Appendix I: Glossary of Terms

Sprint—Is one iteration of the Agile planning and executing cycle. So, the Sprint represents the actual “work” being done on some component of the project and must be completed before the next Scrum meeting.

Scrum—In the sport of rugby, a “scrum” is the restarting of the game following a minor infraction. In Agile PM, Scrum refers to the development strategy agreed to by all key members of the project. Scrum meetings involve assessing the current status of the project, evaluating the results of the previous Sprint, and setting the goals and time-box for the next iteration.

Time-box—The length of any particular sprint and is fixed in advance, during the Scrum meeting and typically vary between one and four weeks in length.

User stories—A short explanation, in the everyday language of the end user that captures what they do or what they need from the project under development. The goal of the user story is to gain their perspective on what a correctly developed product will do for them.

Scrum Master—The person on the project team responsible for moving the project forward between iterations, removing impediments, or resolving differences of opinion between the major stakeholders. The Scrum Master *does not* have to be the project manager but does have a formal role in enforcing the rules of the Scrum process, including chairing important meetings. Scrum Masters are focused solely on the Agile project development process and do not play a role in people management.

Sprint Backlog—The set of Product Backlog items selected for the Sprint, plus a plan for delivering the Sprint Goal. The Sprint Backlog is a forecast by the development team about what functionality will be in the next time-box increment and the work needed to complete that functionality. The team controls the Sprint Backlog.

Burndown Chart—Shows remaining work in the Sprint backlog. Updated every day and displayed for all Scrum members to see, it provides a quick reference of the Sprint progress.

Product Owner—The person representing the stakeholders and serving as the “voice of the customer.” They may be a member of the project organization but must take the “outside,” user’s viewpoint in representing customer needs. The product owner creates **user stories** that identify their specific needs for the product.

Development team—The organizational unit responsible for delivering the product at the

end of each iteration (Sprint). Typically, the development team is cross-functional and self-organizing; that is, they collectively determine the best way to achieve their goals.

Product Backlog—A prioritized list of everything that might be needed in the completed product and is the source of requirements for any changes to be made to the product. The Product Backlog is never “final”; it evolves as the product and the business setting in which it will be used evolve. It constantly changes to identify what the product needs to be appropriate, competitive, and useful. The product owner controls the Product Backlog.

Work backlog—The evolving, prioritized queue of business and technical functionality that needs to be developed into a system.

Citations

Almeida, F. (2017) Challenges in migration from waterfall to agile environments. *World Journal of Computer Application and Technology*, 5 (3), 39-49.

Baseer, K.K., Reddy, M.R.A., Bindu, S.C. (2015) A Systematic Survey on Waterfall vs. Agile vs. Lean Process Paradigms. *i-manager's Journal on Software Engineering*, 9 (3), 34-59.

Beck, K., et al. (2001) The Agile Manifesto. *Agile Alliance*, retrieved 20 December 2020, <<http://agilemanifesto.org/>>

Boehm, B. 1996. Anchoring the software process. *IEEE Software*, 13 (14), 73-82.

Cervone F. (2011) Understanding agile project management methods using Scrum. *OCLC Systems & Services: International digital library perspectives*, 27 (1), 18-22.

Chow, T., Cao, D. (2008) A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, 81 (6), 961-971.

Cooper, R.G. (1990) Stage-gate systems: A new tool for managing new products. *Business Horizons*, 33 (3), 44–54.

Cooper, R.G., Sommer, A.F. (2016a) Agile-Stage-Gate: New idea-to-launch method for manufactured new products is faster, more responsive. *Industrial Marketing Management*, 59, 167-180.

Cooper, R.G., Sommer, A.F. (2016b) From Experience: The Agile–Stage-Gate Hybrid Model: A Promising New Approach and a New Research Opportunity. *Journal of Product Innovation Management*, 33 (5), 513–526.

Dikert, K., Lassenius, C., Paasivaara, M., (2016) Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, 119 (September), 87-108.

Dingsøyr, T., Dybå, T. (2008) Empirical studies of agile software development: a systematic review. *Information and Software Technology*, 50 (9-10), 833-859.

Doshi, N., McGregor, L. (2018) Why Agile Goes Awry — and How to Fix It. *Harvard Business Review*. Retrieved from: <https://hbr.org/2018/10/why-agile-goes-awry-and-how-to-fix-it>.

Fagerberg, B. Verspagen, B. (2009) Innovation studies — The emerging structure of a new scientific field. *Research Policy*, 38, 218-233

Fair, J. (2012) Agile versus Waterfall: approach is right for my ERP project? Paper presented at PMI® Global Congress 2012—EMEA, Marsailles, France. Newtown Square, PA: Project Management Institute.

Fernandez, D., Fernandez, J. (2008) Agile Project Management - Agilism Versus Traditional Approaches. *The Journal of Computer Information Systems*, 49 (2),10-17.

Hobbs B., Petit Y. (2017) *Agile Approaches on Large Projects in Large Organizations* Project Management Institute: Newton Square, PA.

Kisielnick, J., Misiak A.M. (2017) Effectiveness of Agile Compared to Waterfall Implementation Methods in IT Projects: Analysis Based on Business Intelligence Projects. *Foundations of Management*, 9, 273-286.

Laanti, M., Salo, O., Abramsson, P. (2011) Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation, *Information and Software Technology*, 53, 276-290.

Levesque, L., Wilson J., Wholey, D. (2001) Cognitive divergence and shared mental models in software development project teams. *Journal of Organizational Behavior*, 22, 135-144.

Mahalakshmi, M., Sundararajan M. (2013) Traditional SDLC Vs Scrum Methodology – A Comparative Study. *International Journal of Emerging Technology and Advanced Engineering*, 3 (6), 192-196.

Miller, J.R., Haddad, H.M. 2012. Challenges faced while simultaneously implementing CMMI and scrum: a case study in the tax preparation software industry, *Ninth International Conference on Information Technology*, 314-318.

Nichols, G., Lewis N., Eschenbach T. (2015) Determining When Simplified Agile Project Management Is Right for Small Teams. *Engineering Management Journal*, 27 (1), 3-10.

Nils, M., Dingsøyr, T., Dybå, T. (2010) A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and Software Technology*, 52 (5), 480-491.

Pinto, J.K. (2019) *Project Management: Achieving Competitive Advantage*, 5th Ed. Upper Saddle River, NJ: Pearson.

Pirola-Merlo, A. (2010). Agile Innovation: The role of team climate in rapid research and development. *Journal of Occupational and Organizational Psychology*, 83, 1075-1084.

Poppendieck, M., Poppendieck, T. (2003) *Lean Software Development: An Agile Toolkit*. Boston, MA: Addison-Wesley Professional.

Project Management Institute (2017) *Pulse of the Profession*. Newtown Square, PA: Project Management Institute.

Quist C. (2015) *Benefits of Blending Agile and Waterfall Project Planning Methodologies*. University of Oregon Applied Information Management Program – Capstone Report

Readman, J., Squire, B., Bessant, J., Brown, S. (2006) The application of agile manufacturing for customer value. *Journal of Financial Transformation*, 18, 133-141.

Rigby, D.K, Sutherland, J., Takeuchi, H. (2016) Embracing Agile. *Harvard Business Review*. Retrieved from: <https://hbr.org/2016/05/embracing-agile>.

Rogers, E.M. (1983) *Diffusion of Innovation*, 3rd Ed. New York: Free Press.

Scaled Agile, Inc., n.d., SAFe® for Lean Enterprises 5.0, viewed 04 January 2021, <www.scaledagileframework.com>

Schwaber, K., Beedle, M. (2002) *Agile software development with scrum*. Upper Saddle River, NJ: Prentice-Hall.

Serrador, P., Pinto, J.K. (2015), Does agile work? – a quantitative analysis of agile project success, *International Journal of Project Management*, 33, 1040-1051.

Sliger, M. (2006) *A Project Manager's Survival Guide to Going Agile* – Whitepaper – The Rally Software Development Corporation

Sliger, M. (2010) Goodbye, scope creep—hello, agile! Paper presented at PMI® Global Congress 2010—North America, Washington, DC. Newtown Square, PA: Project Management Institute.

Takeuchi, H., Nonaka, I. (1986, January–February) The new new product development game, *Harvard Business Review*, 137–146.

Tendedez, H., Ferrario, M-A, Whittle, J. (2018) Software Development and CSCW: Standardization and Flexibility in Large-Scale Agile Development, *Communications of the ACM on Human-Computer Interaction*, 2, 1-23.

Van de Ven, A. (1986) Central problems in the management of innovation, *Management Science*, 32 (5): 592-609.

Yoon, C. (2018) Assumptions that led to the failure of Google Glass, *Medium*, retrieved at: <https://medium.com/nyc-design/the-assumptions-that-led-to-failures-of-google-glass-8b40a07cfa1e>