1-1-2013

Is a Rigorous Agile Methodology the Best Development Strategy for Small Scale Tech Startups?

Alex Yau
University of Pennsylvania, ayau@sas.upenn.edu

Christian Murphy
University of Pennsylvania, cdmurphy@seas.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-13-01.

Is a Rigorous Agile Methodology the Best Development Strategy for Small Scale Tech Startups?

Abstract

Recently, Agile development processes have become popular in the software development community, and have been shown to be effective in large organizations. However, given that the communication and cooperation dynamics in startup companies are very different from that of larger, more established companies, and the fact that the initial focus of a startup might be significantly different from its ultimate goal, it is questionable whether a rigid process model that works for larger companies is appropriate in tackling the problems faced by a startup. When we scale down even further and observe the small scale startup with only a few members, many of the same problems that Agile methodology sets out to solve do not even exist. Then, for a small scale startup, is it still worth putting the resources into establishing a process model? Do the benefits of adopting an Agile methodology outweigh the opportunity cost of spending the resources elsewhere? This paper examines the advantages and disadvantages of adopting an Agile methodology in a small scale tech startup and compares it to other process models, such as the Waterfall model and Lean Startup. In determining whether a rigorous agile methodology is the best development strategy for small scale tech startups, we consider the metrics of cost, time, quality, and scope in light of the particular needs of small startup organizations, and present a case study of a company that has needed to answer this very question.

Keywords

Agile methodology, Lean Startup, small scale tech startup.

Disciplines

Computer Engineering

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-13-01.

Is a Rigorous Agile Methodology the Best Development Strategy for Small Scale Tech Startups?

Alex Yau and Christian Murphy
Department of Computer and Information Science
University of Pennsylvania
Philadelphia PA 19104
ayau@sas.upenn.edu, cdmurphy@seas.upenn.edu

***Abstract*— Recently, Agile development processes have become popular in the software development community, and have been shown to be effective in large organizations. However, given that the communication and cooperation dynamics in startup companies are very different from that of larger, more established companies, and the fact that the initial focus of a startup might be significantly different from its ultimate goal, it is questionable whether a rigid process model that works for larger companies is appropriate in tackling the problems faced by a startup.**

When we scale down even further and observe the small scale startup with only a few members, many of the same problems that Agile methodology sets out to solve do not even exist. Then, for a small scale startup, is it still worth putting the resources into establishing a process model? Do the benefits of adopting an Agile methodology outweigh the opportunity cost of spending the resources elsewhere? This paper examines the advantages and disadvantages of adopting an Agile methodology in a small scale tech startup and compares it to other process models, such as the Waterfall model and Lean Startup. In determining whether a rigorous agile methodology is the best development strategy for small scale tech startups, we consider the metrics of cost, time, quality, and scope in light of the particular needs of small startup organizations, and present a case study of a company that has needed to answer this very question.

***Index Terms*—Agile methodology, Lean Startup, small scale tech startup.**

I. INTRODUCTION

In recent years, there has been an increased focus on the managerial and organizational aspects of software engineering in tech startups. Software process models are being created and changed constantly with the belief that better process models can ultimately lead to the success of a company. Recently, Agile methodology has become popular in the software development community. Some consider this the best thing that has to the software industry, and perhaps a possible "Silver Bullet" to solve the problems of software development. Over the years, the Agile methodology has proven successful in many large companies [12]. However, we must understand that the communication and cooperation dynamics in startups are very different from that of larger, more established companies and therefore startups may have different problems and concerns that do not apply to giant corporations. Although companies ultimately have the same business goals, "Faster, Cheaper and Better", the initial focus of a startup might be significantly different from its ultimate goal. Depending on the current business environment, the immediate business goal for a startup may change constantly to react to these changes. Therefore, a rigid process model that works for larger companies may be inefficient in tackling the problems faced by a startup.

Furthermore, startups are faced with limitations that their larger counterparts may take for granted. With limited resources and, in many cases, constant direct competition, allocating human resources to defining and maintaining a rigorous methodology is out of the question for some. However, the Agile methodology has been shown as successful in many case studies and research [8]. But is it a "one-size-fits-all"? In tech startups, Agile definitely has clear benefits over some of the other, traditional methods. It has a solid principle and has been shown from past case studies to mitigate certain problems faced by companies. However when we scale down even further and observe the "small scale" startup with, say, only three members, a lot of the same problems that Agile methodology sets out to solve do not even exist. For example, with three members working in a small office, the problem of communication becomes insignificant compared to the problem of communication among a 100 person startup. Then, for a small scale startup, is it still worth putting the resources into establishing a process model? Do the benefits of adopting an Agile methodology outweigh the opportunity cost of spending the resources elsewhere?

This paper examines the advantages and disadvantages of adopting an Agile methodology in a small scale tech startup and compares it to other process models, such as the Waterfall model and Lean Startup, and attempts to answer the question, "Is a rigorous Agile methodology the best development strategy for small scale tech startups?"

II. SCOPE

Before we begin, it is essential to define the scope of the proposed question. Since Agile software development can be considered as merely "a collection of practices, a frame of mind" [6], it is difficult to tell whether a company's process model is defined as Agile. Some may choose to follow those Agile beliefs loosely while others may employ a strict Agile system. Therefore it is important to distinguish companies with different levels of "agility" in order to properly analyze the effectiveness of the agile system. In the scope of this paper, a

rigorous Agile methodology will be defined as one that follows *all* the Agile principles and strict practices, similar to Extreme Programming and Scrum.

Secondly, as briefly mentioned earlier, Agile development may have different effects on a company depending on its stage and size. The cost of implementing the Agile methodology and the benefits vary as the company grows. This paper focuses on the effects of Agile on a "small scale" startup, one composed of roughly eight members or less. This is a good scope to focus on since a majority of startups begin with roughly two to three founding members and perhaps a few more engineers [26]. Thus, the discussion will be mostly concentrated on the cost of implementing the Agile system in a startup of such a scale and the benefits and impact it has in the perspective of the early small scale startup.

Lastly, in order to answer the proposed question and determine if a rigorous Agile methodology is "the best development strategy", we must first discuss the scope of the metrics that we are using to determine the effectiveness of a process model. The metrics used in this discussion are cost, time, quality and scope as they apply to a startup. This paper will thus compare the effectiveness of different process models based on their effects on the four areas mentioned. These metrics will be defined fully in Section IV below.

III. Methodology

This section outlines the necessary steps required to answer the proposed question, "Is a rigorous Agile methodology the best development strategy for small scale tech startups?" We first begin by observing the problems of software development and defining the four metrics – cost, time, quality and scope – and the tradeoffs associated in the perspective of a small scale startup (Section IV). We will then discuss some of the traditional process models, such as the Waterfall model, and their effects on the four metrics (Section V).

The paper will then follow with a detailed definition of the Agile methodology, its principles and the practices associated, such as Extreme programming and Scrum. The impact of Agile methodology on the four metrics will be compared with the traditional process models and their implications on small scale tech startups will be addressed (Section VI).

A popular alternative to the Agile methodology, the Lean Startup will then be discussed and compared to Agile (Section VII). A thorough case study on a small scale tech startup, Everyme, will be presented and used as an example of a Lean Startup that does not employ a strict Agile methodology (Section VIII). The paper ends with a proposed solution to the question raised.

IV. Metrics

Since process models are tools of project management, in order to analyze the quality of a process model, we must first consider the goals of project management itself. According to Olsen's article "Can Project Management Be Defined?", project management is "the application of a collection of tools and techniques...toward the accomplishment of a...task within time, cost and quality constraints" [17]. Similarly, the British Standard for project management [24] defines project management as "the planning, monitoring and control of all aspects of a project...to achieve the project objectives on time and to the specified cost, quality and performance." In the scope of this paper, we focus on the process model as the tool that is used to accomplish a task according to the metrics of time, cost and quality constraints.

Time, cost, scope and quality make up what is commonly known as the iron triangle [2][9] or triple constraints of project management. The iron triangle is a visual representation of the common tradeoffs of project management. It suggests that in order to increase the scope of a project, time and cost must suffer in order to keep the same quality, vice versa. Therefore, by analyzing the impact of a process model on the time, cost, scope and quality of software being developed, we can assess the effectiveness of the model or methodology.

In the scope of software development in small scale tech startups, the four metrics can be defined as:

Time - The total time taken from start of the project to a public release of the product or service

Cost - The total cost spent by the startup, including cost of hiring engineers

Scope - The number of features and extensions (such as language localization) of the product

Quality - This includes both internal quality, such as testability and maintainability, and external quality, such as usability and reliability

V. Traditional Process Models

One of the more popular traditional process models is the Waterfall model. The Waterfall model has been around since the 1970s and is "a framework for software development in which development proceeds sequentially through a series of phases" [14]. The progress flows from one phase to another in order, although short feedback loops are allowed. It is possible to move backwards and make modifications based on the feedback, but other than that the system generally follows these distinct steps:

1. Requirements analysis - The first step is to gather information, define the scope and understand and analyze the specifications of the project.

2. Design - The second step is to define the hardware and/or software architecture, modules, interfaces, etc. to satisfy the requirements specified in the first step.

3. Implementation - This step consists of actually coding and constructing the software based on the design and requirements established from the previous two steps.

4. Testing - In this step, all the components are integrated together and tested to ensure they meet the customer's specifications as specified in the first step.

5. Installation - This step prepares the product for delivery for commercial use.

6. Maintenance - The last step involves making modifications to improve the quality and performance on the system based on the feedback from the customer.

From the outline of the Waterfall model, we can see some immediate benefits to software development in startups. The model provides a clearly defined structure that enforces discipline for a startup. It provides a clear direction with a transparent way of assessing progress through the use of milestones. Since a direction is not immediately obvious to young startups, and the software development process may be quite unstructured and unorganized, the Waterfall model can not only provide a clear vision and goal for the startup but also a clean software development structure through the use of stages.

The Waterfall model also puts a huge emphasis on customer specification analysis, the first step in the model, and the design structure of the software even before the team starts writing code. If done correctly, this can reduce both cost and time in the software development phase as it minimizes the time and effort wasted on writing code that does not meet customer specifications or constantly refactoring because of bad code design.

Lastly, the Waterfall model may improve the overall quality of software since flaws in the design and misunderstanding of specifications are handled in the first two steps before the code is written rather than trying to catch those mistakes in the testing stage. Furthermore, since all specifications and design architectures are properly documented after the first two stages, communication time between team members can be greatly reduced.

However, since this model relies on the customer specifications being clearly defined in step one, the specification documents created in the first step may become outdated if the customer changes his mind. In startups, the vision and the scope of the product are usually not fully formed and thus customer specifications may change drastically from one day to another. Since the phases of the Waterfall model are built on top of each other such that the design phase follows the specifications defined in step one and the implementation stage depends on the design structure, a lot of time may be wasted if specifications change. This problem is amplified especially in startups because their scope tends to change constantly to adapt to the needs of the customer (or the market) and the need to refine their product. As a result, the cost and time may increase drastically in some cases.

Therefore, unless the specifications are clearly defined and unchanging, which is rare in a startup, the Waterfall model may be more detrimental than beneficial to a small scale tech startup.

VI. AGILE

Agile methods are a reaction and a proposed solution to traditional methodologies like the Waterfall model that acknowledge "the need for an alternative to documentation driven, heavyweight software development processes" [8]. In fact, according to Cockburn and Highsmith, Agile software development does not necessarily introduce new practices but is the "recognition of people as the primary drivers of project success, coupled with an intense focus on effectiveness and maneuverability" [7].

At its core, the Agile methodology focuses on incremental and iterative development similar to the spiral model. It aims to avoid detailing and defining the entire project at the beginning like the Waterfall model, but instead to plan out and deliver small parts of the project at a time. The methodology is similar to having small loops of the Waterfall model for each feature in the software. The development process starts with the most basic set of deliverables, followed by planning, implementing and testing the next set of features in subsequent iterations. The purpose of this development process is to increase the agility of the development team, by minimizing the time and cost wasted if the customer decides to change his mind.

According to Cohen, Lindvall and Costa, being Agile "involves more than simply following guidelines that are supposed to make a project Agile" [8]. Andrea Branca also states that some "processes may look Agile, but they won't feel Agile" [6]. However, there are some methodologies and processes with such a great emphasis on Agile beliefs that they can be considered the core of Agile methodology and have been widely adopted by top companies in the world. This paper will now explore a few of these Agile methodologies and discuss their effectiveness in a small scale startup.

A. Scrum

Scrum, first introduced by Ken Schwaber in 1996, is a widely used Agile methodology that focuses on developing software in short iterations known as sprints. The process consists of the following stages:

Pre-sprint planning - Features and functionalities are selected from a backlog, and a collection of features are planned and then prioritized to be completed in the next sprint.

Sprint - The team members choose the features they want to work on and begin development. Scrum meetings are held daily, every morning, to aid communication between developers and product managers. A sprint usually last between one to six weeks.

Post-spring meeting - In this meeting, the team analyzes the progress in the past sprint.

In the perspective of a small startup, this process provides a couple of benefits. The enforced daily meetings can improve communication between team members. This can not only decrease the time and cost due to possible miscommunication otherwise, but can also improve the quality of software since

software can be better designed when each member understands the overall scope of the project and how others are implementing certain parts. Since the overall structure of the software changes much faster in a startup than in a larger company, it is necessary to keep everyone updated in order to achieve good quality of software.

On the other hand, the pre-sprint planning helps the team narrow down their to-do list and focus on the immediate goal. This is particularly important to startups because the final product is not fully defined and thus it is easy for developers to fall into the trap of developing too many features instead of concentrating on the main features. Therefore, by imposing a constraint of time with short iterations, the process helps the team focus on its goal and deliver the necessary features.

However, although a constraint of time in Scrum and Agile can narrow the focus and discourage startups from implementing unnecessary features, some may argue that this process harms the scope of the project and limits the creativity that is important in a startup. Iterative development of prioritized features with a time constraint discourages the development team from exploring different ways to implement a certain feature that may perhaps be more efficient or provide more value to the project. Since it is difficult for startups to break into an existing market, innovative designs and implementation of features are particularly important in determining the success of a startup. Therefore, a startup must consider the tradeoffs of scope and creativity to time and cost when thinking about adopting a more focused and iterative development process.

B. Extreme Programming

Extreme Programming is another methodology that encompasses the core concepts of Agile development similar to Scrum. In "Extreme Programming Explained: Embrace Change", Beck outlined the 12 rules of Extreme Programming [3]. In addition to the rules mentioned in Scrum, like the focus on pre-iteration planning, short releases and simple design, Extreme Programming also encourages other Agile practices.

Extreme Programming encourages test-driven development and suggests that the developers write acceptance test for their code before they implement the features. The benefits of test-driven development are clear: writing test cases before implementing a feature can ensure the feature fulfils the specifications that were set out originally. Furthermore, the quality of software is also improved not only because of the decrease in bugs and faults in the software but also because of improved maintainability of the software. With tests written for all the features implemented, it is easy to tell whether changing a section of the code is going to affect another section simply by running the test suite. Therefore, test-driven development can definitely increase the quality of software and decrease the cost and time wasted on debugging afterwards.

However, do the same benefits apply to a small scale startup? A small scale startup has a limited number of developers, a list of features that is probably being changed and refined constantly and a limited amount of time and money. Is it worth spending time writing comprehensive test cases for every feature before implementing it? It is very possible that by the time the tests were written, the customer has changed his mind and the tests will be rendered useless. On the other hand, if the same amount of time has been spent on developing the feature, the code may be recycled for another feature.

Furthermore, in many cases, the customer may request a few features as prototypes to test out some ideas in order to make up his mind. When that happens, it does not seem reasonable to write out all the tests but instead it would be preferable to implement those prototypes as fast as possible in order to speed up the decision and design process.

Lastly, a small scale startup that has not obtained much funding will probably have a short runway, and thus a limited amount of time and money. The priority in this case will be to create an MVP, or minimal viable product, which may lack in quality but is at least functional enough to pitch to and show investors.

Overall, the test-driven development aspect of Agile is a tradeoff between cost and time to achieve improved quality of software. Although quality is important, as startups usually only have a few chances to make a strong impression on investors and users in the market, cost and time may be a larger deciding factor. Once the startup runs out of funding or if a close competitor releases a similar product, a higher quality of half a product isn't going to help much.

The Extreme Programming process also places emphasis on pair programming, a process that requires two developers to write code together on the same machine. This is often used with the purpose of creating better written code, increasing discipline and emphasizing collective code ownership [13]. The idea is that paired programmers are less likely to take longer breaks and are more likely to "do the right thing" under someone else's watch. Pair programming can also allow the programmers to bounce ideas off each other and thus be less likely to overthink a simple problem or to reach a programmer's block. It also encourages collective code ownership by increasing a programmer's knowledge of the code base through pairing with different programmers. The benefits listed above can again increase quality of the software at the cost of money and time. In addition, pair programming usually provides a good morale boost within the team and is often used in large companies due to the benefits it provides to the project management of large teams.

However, for a small scale startup with fewer than eight employees, the benefits of pair programming may be limited. As discussed earlier, small startups have a tight constraint of time and money, and thus improving quality with twice the cost (of hiring two developers) may be out of scope for a small startup. Furthermore, since the team is very small, each developer is probably responsible writing code in different areas of the code base, and thus already reaps the benefits of collective code ownership advertised by pair programming.

A common system of assessing progress and defining features in Agile methods such as Scrum and Extreme Programming is the use of user stories, velocity and backlogs [19]. A user story is a description of a feature in everyday language that can be easily understood by non-technical persons. For example, a user story can be "As a user, I want to

be able to log into the site with my Facebook account". Each user story can then be assigned points based on the time it takes for the feature to be implemented as estimated by the developer. The velocity of the team can be calculated as "the sum of the time estimated of user stories implemented within an iteration/release" [11], in other words, the sum of the points given to the user stories. By describing features in a non-technical language, the system encourages the integration of business and marketing to the implementation of the product which can improve the usability of the software. The use of velocity to measure the team's progress can also provide a quantitative assessment to the project manager and can help estimate the features that can be delivered before a certain deadline.

Other than velocity, there are other metrics that are used to assess a team's performance, such as defect rates, defined as the number of defects made by a team and by each programmer during each iteration.

However, are these metrics that important to a small scale startup? From the above discussion, we can definitely note the importance of the Agile process and methodologies in software development. As a good "tool for project management", it proves to be beneficial to improve quality and decrease time and cost of the project while keeping it in scope and focused in projects with a large team. The Agile methodology is a well established system that can also act as a guide and provide a good structure for startups that do not have a clear plan for managing their team and analyzing the progress.

However, we remain skeptical of whether small scale startups can actually reap the full benefits of following a rigorous Agile process. Agile may be popular in the startup world, but startups that are in a much earlier stage, with much fewer employees, are beginning to favor a relatively new process model called Lean Startup. Perhaps this new way of project management is more lightweight and better suited to the bootstrapping style of these early small scale startups.

VII. Lean Startup

Lean Startup, a term coined by Eric Ries [21], is a process model that "builds on many previous management and product development ideas, including lean manufacturing, design thinking, customer development, and agile development". Although the Lean Startup process does involve some core principles of Agile methodologies discussed earlier, the main difference between Lean Startup and Agile is that Lean eliminates anything that is not absolutely necessary, including possibly team meetings, tasks and documentation.

It is important to note that Agile and Lean are not mutually exclusive, but rather largely complementary. In "*The Lean Startup*" [21], Ries emphasizes the importance of learning in the process. Lean Startup focuses on learning how to build a sustainable business, whether to pivot or preserve, and entrepreneurial management. According to Ries, it is important to distinguish whether the outcome of a startup's effort is value-creating or wasteful. For example, since customer specifications change all the time, learning to gain important insights about customers contains much more value in the long run than focusing on making the product better by adding features and fixing bugs based on what the customers want at the time.

Ries argues that although Agile development methodologies were designed to eliminate waste by decreasing the duration of feedback loops, a lot of waste still occurs because of mistaken assumptions. Agile as well as the "Lean thinking" in lean manufacturing defines value as effort that "[provides] benefit to the customer" [21]. However, who the customer is, what the customer wants and what the customer may find valuable are unknown and subject to change. Therefore Ries proposes that the value of a startup should arise from the effort spent on learning about "what creates value for customers".

The majority of Agile methodologies include techniques to aid in project management and progress analysis, such as the use of user stories, backlog and velocity, and also on finding the most efficient way to build features and make corrections that satisfy the customer's current decisions. On the other hand, Lean Startup focuses on validated learning as the metric in measuring progress and value. For example, the Agile methodologies employ acceptance testing, tests that are based on customer specifications, as the testing strategy while Lean Startup advertises the use of split testing (an experimental approach that tests two variations of the software). The Lean Startup methodology believes that a startup only assumes who their customer is, but does not know exactly what the customer wants and what their final product should be. Through using validated learning methods such as split testing, the startup is able to learn more about the customer and be able to make decisions, learn and improve their product in a way that is meaningful.

Similarly, the release log and backlogs in Agile build toward a release plan while Lean Startup works towards deploying a minimal viable product. It is this continuous deployment and validation that provide startups with knowledge of their market and customers. Although focusing on building a minimal viable product may sacrifice the quality of software in the short term, the startup benefits from the decrease in overall development time and cost. Through continuous deployment and validation, startups are able to push out products very quickly and continue to improve their quality in the longer run. Furthermore, the scope and quality of their software ultimately benefits in the long run due to the focus on learning and customers.

Ultimately, Agile methodologies tend to target the actual development of software while Lean Startup is more beneficial to business development and product management. In a small scale startup, perhaps the benefits gained by improving business and product development are more important in the long run than improving the actual process to develop the software behind it. In the next section, we will take a look at a typical small scale startup that has employed a mix of both Agile and Lean process models to observe both process models in practice.

VIII. Case Study: Everyme

In the Summer of 2012, the first author had the opportunity to intern at Everyme, a Y-Combinator startup that was building a private social network app for friends, families and partners. At the time of employment, they had a team of five, which falls under our definition of a small scale startup. The team consisted of a designer, an iPhone developer, an Android developer and a web developer (who is also a co-founder); the CEO also worked on iOS development from time to time. This is a typical setup of a small tech startup, with very small teams of one to two working on their part. At the time of the internship, Everyme was using a process model containing the elements of both Lean Startup and Agile. The CTO of the company, Vibhu Norby [16], had also described the effects of their process model on the management level during an interview.

From a developer's perspective, the model was fairly simple. Since everyone was basically "in their own department", they worked at their own pace and prioritized work in their own way. Occasionally, integration across the mobile and web platforms was needed and tasks needed to be reprioritized. There was a five minute stand-up meeting once every few days to go over what each person had done previously, what he would do next and whether he would need anything from anyone. For example, an Android developer may ask the designer for templates. This provided everyone with a rough idea of company's progress and whether they had to re-prioritize their list of tasks. This is similar to the Agile process Scrum's pre-sprint stand-up meeting, but more casual and without coming up with a list of tasks that are required to be completed by the week (or sprint). Everyme did not employ a backlog system but instead had an issue list for people to assign certain tasks to each other. This gave flexibility to each developer to work on something he was interested in and provided the developers with more room for innovative implementations and features that may not necessarily be in the backlog. This is very similar to the Lean Startup ideology.

However, Norby argued that this process did bring some disadvantages. Without an Agile-esque backlog or a required list of tasks that needed to be completed by a certain time, there were times when the developers were unclear of what to do. When a huge decision or possible pivot was being discussed and formed, which was more often in a small startup than in a large company, the direction of the project became unclear and thus time and cost were wasted as developers were unsure of what they need to do.

Everyme also used a Lean Startup approach when it came to assessing progress of the team and company. Instead of using the difficulty and number of features done per week similar to velocity in Agile, Everyme used a validated approach, based on the number of downloads, the reviews and feedback they received and so on. Milestones and inflection points were also used to observe the general progress of the company. It was found to be much more effective as a motivational tool to set up inflection points, such as a release date, or important dates, such as meetings with investors that required the product to be done. People performed better under constraints. However, Lean Startup's validated approach may only be beneficial up to a certain point. Norby noted that "progress measured by downloads, as done in Lean, may not be effective in the long run. You can have up to millions of downloads, but that doesn't tell you which direction to go next."

When asked about whether they have tried adopting a more rigorous Agile methodology, Norby described that they have tried using Sprint.ly [23], an online system that uses the Scrum process. Similar to Scrum, it defines tasks as user stories with a certain difficulty. These tasks are then stored in the backlog and taken out when a developer decides to implement it. However, implementing Sprint.ly into their current process model was too costly and time consuming. For a small scale startup like Everyme, everyone has his own process model and work schedule. Employees have their to-do list in their mind and they all know roughly how long it will take. Spending time writing it down, modifying it and crossing it off later is just too unnecessary.

Norby went on to describe how they had tried test-driven development, another important aspect of the Agile process, but found that it was also too time consuming. "We would spend a lot of time writing test cases for features that may end up not being implemented, because you know, specifications change all the time."

For small scale tech startups similar to Everyme, we can see that although the Agile principles are important, they may be too costly to implement. "We don't even have a project manager… it takes too much effort for us to take time out of our schedule to manage this". With a small enough team that functions well without management, it may seem unnecessary to insist upon a strict process model. Therefore, a combination of the Lean Startup approach and Agile principles may mitigate the problem by having less of a structured process but still provides the benefits that Agile proposes.

IX. Related Work

There has been much research in the past that considered the suitability of Agile processes to various software organizations, but this prior work does not consider the challenges of small scale startups in particular [1][20] and/or does not address the impact of the process on the metrics of cost, scope, time, and quality [5][22], as we do here.

Others have assessed various aspects of Agile software development (e.g., pair programming [15] or test-driven development [4]) but have not related the overall effect in a small startup environment.

Additionally, some researchers have investigated the combinations of Lean Startup and Agile [25], and the tradeoffs between Agile and traditional approaches [18], while others have compared the two when used in a startup [10], but we believe that we are the first to specifically address the issues related to Agile processes and a small scale startup company of eight or fewer employees.

X. Conclusion

We have discussed process models such as the Waterfall model, the Agile methodologies, Extreme Programming and Scrum, and Lean Startup and their effects on small scale tech startups. We also looked at the effects of these process models on a startup in practice. Through our discussion, we concluded that different process models have different tradeoffs between the four metrics – cost, time, quality and scope – and are most beneficial when employed during the different stages of a company.

While the Waterfall model is effective for companies with a solid, unchanging end goal, it performs badly with startups that are unsure of their final products. The model can help companies decrease the cost and time of development by defining the specifications and design architectures at the beginning but suffers when specifications change drastically. This may be beneficial to large companies with unchanging goals, but becomes ineffective for startups, which are likely to be unsure of their end goals and may change their specifications constantly.

The Agile methodologies attempt to solve this issue by decreasing the feedback loops by integrating customer feedback to the development process. Tasks are translated into user stories, a format understood by business persons, in order to aid communications between business and development. Unlike the Waterfall model, customer feedback can then be easily integrated into the development process and the startup is able to make changes easily with minimal waste of time and money. A rigorous, purely Agile process model can no doubt increase the quality of the software, but at a cost of extra time and money required to manage and maintain the system. At a hundred person startup or even a large established company, the cost of maintaining such a system is fixed and spread out, making the tradeoff of quality against time and cost worth the implementation of an Agile process model. On the other hand, the fixed cost and time of implementing a similar system in a small scale startup may be too high for the quality gained.

The Lean Startup model largely complements the Agile methodologies but argues that the Agile way of using velocity, the difficulty and number of features implemented in each iteration, is a poor indicator of progress and suggests the use of validated learning as a process model to determine the progress of a company. The Lean Startup methodology observes the excessive amount of process in the Agile model and attempts to mitigate the problem by decreasing the number of rigorous practices in a startup to strike a balance between quality, time and cost that is suitable for a small scale startup.

In a small scale startup at Everyme, we saw that although the Agile methodology does provide a good process for managing teams of large sizes, a small scale startup may not experience the same problems as a large company and thus may not reap the full benefits of adopting an Agile methodology. While it is important to understand the Agile principles so the team does not fall into the trap of premature optimization and planning similar to the Waterfall process, a rigorous process may be too costly for a startup. Many Agile practices, such as test-driven development and pair programming, provide increased quality of software at an expense of cost and time. Furthermore, a heavy process model may in fact limit the scope of the project by discouraging innovation through a strict backlog or to-do list.

Thus, when considering whether a rigorous Agile methodology is the best development strategy for a startup, we have to consider the different tradeoffs of cost, time, quality and scope. For a small scale startup containing fewer than eight members, a rigorous, purely Agile methodology may not provide enough benefits to outweigh the cost and time put into implementing and managing the process model. It is definitely important to understand Agile principles but perhaps following the Agile methodology strictly is out of scope for a small startup.

Thus, to answer the question, "Is a rigorous agile methodology the best development strategy for small scale tech startups?", we have to determine the ultimate goal of the startup. In general, the Agile process model is most beneficial to improving the process of software development while Lean Startup is most beneficial to business and product development. A startup that is developing software for another company may already have a clearly defined product and does not have to worry about business development. In such a case, a rigorous, purely Agile approach will be most beneficial. On the other hand, if the startup is in charge of the business and product development, or when the software plays a huge part in the product, a hybrid of Agile and Lean may provide the most benefits in terms of the four metrics.

Ultimately, a process model should be transparent enough to allow the team to know how the company is doing but at the same time not burden the developers and allow them to concentrate on what they do best. In a startup, the developers tend to be more invested and interested at the product that they do not require a strict to-do list or motivational benefits from the Agile methods.

A possible area for future research is the analysis of the effects of process models on mobile-centric startups. Practices such as continuous integration in Agile or continuous deployment in Lean Startup become nearly impossible in a startup with heavy focus on mobile development. Since iOS apps have to get approved by Apple, the deployment process usually takes around a week. Even then, a startup cannot force its customers to upgrade to the newer version straight away, unlike web applications. Then, the process models that focus heavily on the ability to integrate and deploy continuously or split testing may not be effective for mobile-centric startups. In this new era in which mobile development is becoming more and more popular, perhaps a new process model is required.

Acknowledgment

The authors would like to thank Kristin Fergis for her initial investigation into this topic, and Vibhu Norby for providing insight into Everyme's organization and processes.

References

[1] S. W. Ambler, "Lessons in agility from Internet-based development," IEEE Software, vol. 19 no. 2, Mar/Apr 2002, pp. 66-73.

[2] R. Atkinson, “Project Management: Cost, Time And Quality, Two Best Guesses and a Phenomenon, Its Time to Accept Other Success Criteria”, International Journal of Project Management 1999, vol. 17 no.6, pp. 337-42

[3] K. Beck, “Extreme Programming Explained: Embracing Change.” Addison-Wesley, 1999. Print.

[4] T. Bhat and N. Nagappan, “Evaluating the efficacy of test-driven development: industrial case studies,” Proc. of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, 2006, pp. 356-363.

[5] J. Chong, “Social behaviors on XP and non-XP teams: a comparative study,” Proc. of the 2005 Agile Conference, July 2005, pp. 39-48.

[6] A. Cockburn, “Agile software development joins the ‘would-be’ crowd.” Cutter IT Journal, vol 15, no. 1, 2002. pp. 6–12.

[7] A. Cockburn and J. Highsmith “Agile software development: The business of innovation” Computer, vol 34 no.9 2001. pp.122

[8] D. Cohen, M. Lindvall, P. Costa. “An Introduction to Agile Methods.” Advances in Computers vol 62, 2004

[9] A. De Wit, “Measurement of Project Success”, International Journal of Project Management 1988, vol 6, no.3, pp. 164-170

[10] J. Dorette Jacob, “Comparing Agile XP and Waterfall software development processes in two start-up companies,” Master’s Thesis, Chalmers Univ. of Technology, Göteborg, Sweden, Nov. 2011.

[11] S. Ilieva, P. Ivanov, E. Stefanova. “Analyses of an agile methodology implementation.” Euromicro Conference, 2004. Proceedings. 30th pp.326-33

[12] P. Krill, “Agile software development is now mainstream,” InfoWorld, 2010, [Downloaded: January 31, 2012.] http://www.infoworld.com/d/developer-world/agile-software-development-now-mainstream-190.

[13] T. Machinnon. “XP: Have you got the discipline?” TickIt Internation magazine. Issue #2Q04, 2004

[14] Melonfire. “Understanding the pros and cons of the Waterfall of software development.” Tech Republic, 2006. [Downloaded: January 31, 2012.] http://www.techrepublic.com/article/understanding -the-pros-and-cons-of-the-waterfall-model-of-software-development/6118423

[15] F. Padberg and M. Müller, “Analyzing the cost and benefit of pair programming,” Proc. of the Ninth International Software Metrics Symposium, Sept. 2003, pp. 166-177.

[16] V. Norby, Personal Interview. 24 Aug. 2012.

[17] Olsen, Richard P. "Can Project Management Be Defined?" Project management quarterly, vol. 2, no. 1 Mar. 1971, pp. 12-14

[18] K. Petersen and C. Wohlin, “A comparison of issues and advantages in agile and incrementeal development between state of the art and an industrial case,” Journal of Systems and Software, vol. 82 no. 9, Sept. 2009, pp. 1479-1490.

[19] M.J. Rees, “A feasible user story tool for agile software development?” Software Engineering Conference, 2002. Ninth Asia-Pacific 2002 pp. 22-30

[20] D. J. Reifer, “How good are agile methods?” IEEE Software, vol. 19 no. 4, Jul/Aug 2002, pp. 16-18.

[21] E, Ries. “The Lean Startup, How Today’s Entrepreneurs Use Continuous Innovation to Create Radically Successful Businessess” Crown Business. 2011. Print.

[22] O. Salo and P. Abrahamsson, “Agile methods in European embedded software development organizations: a survey on the actual use and usefulness of Extreme Programming and Scrum,” IET Software, vol. 2 no. 1, Feb 2008, pp. 58-64.

[23] Sprint.ly (2012) From Sprint.ly Inc website https://sprint.ly

[24] “The British Standard for project management” 6079. ISBN. 0 580 25594 8.

[25] X. Wang, “The combination of Agile and Lean in software development: an experience report analysis,” Proc. of the 2011 Agile Conference, 2011, pp. 1-9.

[26] YCPages (2012) http://ycpages.info/ [Download: February 2, 2013]