DOI NUMBER: 10.19255/JMPM01306

AGILE PROJECT

KEYWORDS
Agile • Practices • Project success • Success factors • Project management • Software projects • Software teams

• ABSTRACT •

In project management, and particularly software project management, there has been a shift from traditional plan based project management, to the agile event driven project management style. This paper identifies some of the most important agile practices a team should use, to succeed in an agile software project. Four participants in two different projects were interviewed. 53 often used practices were identified. 15 were found to be especially relevant in agile software projects. Six practices were related to quality, eight were related to scope and one was related to time. The results indicated that practices which improves customer feedback, helps the team to understand customer needs and improves the team process, are most likely to affect project success.

ANDRE HENRIKSEN
• The Arctic University of Norway
• *andre.henriksen@uit.no*

SVEN ARNE R. PEDERSEN
• Tromsø University Business School
• *sven.arne.r.pedersen@uit.no*

A qualitative case study on

AGILE PRACTICES AND PROJECT SUCCESS

in agile software projects

1.INTRODUCTION

Agile project management is becoming the new de facto standard for developing software. More and more companies are using agile to deliver software faster and in a smarter way (VersionOne, 2016). Agile was originally developed for software projects, but because of the potential benefits, it is now used in other types of projects as well (Serrador & Pinto, 2015).

Dybå and Dingsøyr did a systematic review of agile software development in 2008 (Dybå & Dingsøyr, 2008). The conclusion was that, although a lot of research has been done on agile software development and several benefits and limitations have been identified, the strength of this evidence is very low. According to the review, more high quality studies in agile software development are needed.

Software projects date back as far as the late 1960s. The software industry grew fast and computer companies saw the potential in software production, which had a low cost compared to hardware production and circuitry which were more common. Software companies adopted the already well known waterfall model for its software projects. It turned out that this linear approach for developing software was less than optimal (Mens, 2008). The inflexible separation of phases and the fact that requirements are not always clear at the start of a project, were two major limitations of this model. The main causes for software project failures were: *unrealistic project goals, poor estimates, badly defined requirements, poor status reporting, unmanaged risk, poor communication, use of immature technology, high project complexity, poor development practices, poor management, stakeholder politics* and *commercial pressures* (Charette, 2005).

A new approach was needed and several new lightweight methods started appearing in the late 1980s and throughout the 1990s. These methods advocated an *iterative* and *incremental* approach to software development. This was meant to facilitate a closer collaboration with the customer by encouraging changes throughout the project, to better support customer needs. In 2001, representatives from several of the most important lightweight methodologies met to discuss and find common ground. They formed the Agile Alliance, and the Manifesto for Agile Software Development

(Agile Alliance) was created. They defined four values all agile methodologies must conform to. They also created 12 principles which should be used as guidelines when running agile projects.

The number of failing or contested software projects each year is high. In the latest CHAOS Report it is reported that "only 16.2 percent of software projects are completed on-time and on-budget" (The standish Group, 2015). The size of the company greatly affects this value. In large, medium and small companies, only 9 percent, 16.2 percent and 28 percent of projects are considered a success, respectively. The reports further indicates that agile projects are more likely to succeed, compared to projects based on traditional methods like waterfall (The standish Group, 2015). Agile knowledge and agile usage is increasing. The 10th State of Agile Survey (VersionOne, 2016) indicates that 45 percent of respondents are using agile in most of their projects. 95 percent of the organizations in the survey report some usage of agile development. Scrum and Scrum hybrids are the preferred methodologies, used by 74 percent of agile teams.

A common view in the software industry is that by using an agile project management style, a software project is more likely to succeed. Even though the strength of the empirical evidence is low, agile has for some time been perceived as something that will revolutionize software development and software projects (Dybå & Dingsøyr, 2008). A recent study on agile project success suggests that there might be some truth in this. Serrador and Pinto (2015) conducted a large-scale quantitative study to test if using agile methods had an effect on project success.

They found indications that the use of agile methods correlated to a higher reported success rate. This was shown for three categories of success: *overall project success, efficiency* and *stakeholder success* (Serrador & Pinto, 2015).

Agile projects are characterized as having a more flexible process compared to traditional projects. This process is made of a set of practices, which describes the routines the project team are using to achieve the project goals. For software projects, which are projects where the goal is to create a working software product, this agile approach assists in defining the project scope throughout its lifetime. Scope, as well as time, cost and quality, are important criteria when considering the success of a software project.

In this paper the focus will be on agile practices and trying to determine which practices are most important to achieve project success. The aim is to investigate how agile practices can contribute to increase project success in small software projects.

The rest of the paper is divided into four sections. In section 2 a literature review on agile success factors and agile practices is conducted. In section 3 the methodology will be described. In section 4 an analysis of the findings will be presented and examined. The conclusion is found in section 5.

2. LITERATURE REVIEW

The traditional approach for defining success in a project is to use the project management triangle where *scope*, *time* and *cost* each form a side of a triangle. Making changes to one of the element in the triangle will affect the others (Atkinson, 1999). Quality is often included as a separate element (iron triangle) and in software projects, quality is considered very important (Chappell, 2013). Quality can be defined in several ways, but for software products, there are often three types of quality: *functional-*, *structural-* and *process quality* (Chappell, 2013). *Functional* quality refers to how well the product works for the intended user.

Structural quality refers to the product's source code quality. *Process* quality refers to how the system is created and the process around it.

Traditionally, for a project to be considered successful, it must be delivered on time, within budget and with all the required features and functions. However, different stakeholders involved in a project might have different views on what constitutes success. While external stakeholders usually look at cost and time, internal stakeholders often use scope and quality as the most important criteria for determining success (Agarwal & Rathod, 2006; Bryde & Robinson, 2005). Wicks and Roethlein (2009) consider customer satisfaction the most important part of quality. This is supported by the 10th annual State of Agile survey (VersionOne, 2016) where customer/user satisfaction was ranked third on how success is measured. A recent study by Siddique and Hussein (2016) also shows that customer satisfaction is considered an important success factor, especially in agile projects.

A success factor is defined as something that "must go well to ensure success" (Boynton & Zmud, 1984). In this context, success relates to the outcome of a software project. Several high-level success factors were identified as part of this review.

Lindvall et al. (2002) conducted an online workshop with eighteen agile experts from around the world. One of the goals of this workshop was to identify agile success factors. The three most important success factors they found were: *culture, people* and *communication*. Chow and Cao (2008) conducted a survey among agile professionals from 25 countries. Three critical success factors and three possible success factors were identified: *delivery strategy, agile software engineering techniques, team capability, project management process, team environment* and *customer involvement*. Misra et.al. (2009) did a large-scale empirical study to identify agile success factors. They found nine success factors which were shown to be statistically significant: *Customer satisfaction, customer collaboration, customer commitment, decision time, corporate culture, control, personal characteristics, societal culture,* and *training and learning*.

These three articles indicate a lot of success factors, but there is also a lot of overlap. After narrowing the list down to factors which are likely to be affected by which agile practices are being used in a project, the following list was identified: *communication, engineering techniques, project management process, people and team, customer involvement and satisfaction,* and *decision time*. Three additional papers were found which focuses on agile practices.

William et.al. (2011) did a case study on three Microsoft Scrum teams. They were able to show that by combining an agile framework (Scrum) and nine additional practices, the teams could improve quality, productivity and estimation accuracy. A systematic review on agile practices was conducted by Jalali et.al. (2012) to determine the status of combining agility with engineering practices. 81 peer-reviews articles were identified, from which 61 were empirical studies. 53 of these described a successful agile implementation and were finally included in the study. The State of Agile Survey, by VersionOne, is an annual survey where thousands of respondents participate each year. It has been running for 10 years, which makes it «the largest and longest running agile survey in the world» (2015). There are a lot of agile practices available and some are more popular than others. In the latest State of Agile Survey (VersionOne, 2016), a list of the 25 most commonly used practices were identified.

Project Management traditionally consists of five process groups: *Initiation, Planning, Executing, Monitoring & controlling* and *Closing* (Project Management Institute, 2013). In traditional project management, these are done in a linear and incremental fashion. In agile project management these are done in a more iterative and adaptive way (Wysocki, 2014). Agile methods are both *incremental* and *iterative*. Incremental because the work (scope) is pre-divided into smaller batches of work, and iterative because the scope of each batch is defined just before the start of each loop. This iterative approach makes the process very flexible. **Figure 1** illustrates this looping nature. When the project starts (initiation), the scope is defined. During each iteration, part of the scope is selected (planning) for implementation. The scope subset is then implemented (executing), tested (monitor & control) and closed. If the full scope is not yet implemented, a new iteration is started.

Different agile methods have different practices. When choosing a method for a given project, these practices should be considered to make sure they fit with the project. A method can be viewed as a collection of best practices, values and/or principles, which has been proven to work for certain types of projects. *Scrum* describes practices in form of events, roles and artefacts (Griffiths, 2012). *Extreme programming (XP)* describes 24 practices in form of engineering practices (Beck & Andres, 2004). *Kanban* describes five core principles which also could be viewed as practices (Griffiths, 2012). Various Scrum hybrids exists. *Scrumban* is a model based on Scrum and Kanban which combines the roles, events and artefacts of Scrum and the Kanban board with its work-in-progress limitation. *Scrum/XP* is a combination of practices and rules of Scrum and XP (Mar & Schwaber, 2002). Several other agile methods exist. These are just a few examples on how different methods and frameworks have different practices. In total 53 practices were identified during the literature review. These practices are listed in **Appendix A**.

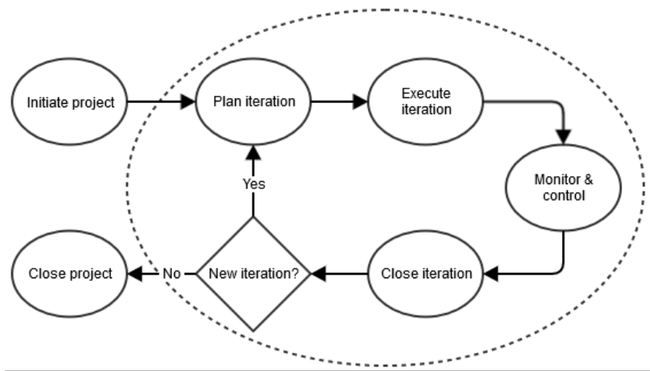

FIGURE 01. Iterative model

3. METHODOLOGY

Focusing on only one level of analysis, trying to describe a phenomenon in its real-world context, while at the same time trying to achieve a degree of literal replication, this paper is based a holistic multiple-case descriptive case-study (Yin, 2014). An open-ended interview was done with four respondents. One team from two different Norwegian software companies were interviewed. In each team one respondent had the role of *product owner* and one was a software *developer*. In both teams, the product owner was located in Norway, while the developer was located in Sri-Lanka, as part of an outsourcing partnership. There are only four respondents in this study, but efforts were made to include team members with different perspectives.

The respondents were interviewed one by one. One respondent was interviewed face to face. Three respondents were interviewed via Skype using a video feed. Two candidates were interviewed in Norwegian while two were interviewed in English. All interviews were recorded (audio only), translated into English and transcribed. An open-ended interview style was used i n order to avoid interview bias and capture the actual thoughts of the respondent. Follow up questions were done via Skype chat and e-mail. To anonymize the respondents, the *product owners* will be referred as PO1 and PO2. The *developers* will be referred to as D1 and D2. The two projects will be referred to as P1 and P2.

Based on the research goal and research done in the literature review, an interview guide with seven questions was created. The guide has two sections. In the first section, questions related to the company, candidate and case under study were defined. The second section contains questions related to the agile practices used in

the case under study. First in section two there is an open discussion where the candidate explains how the team worked together based on the agile method and the practices they used. The goal is to get an in-depth account of the project process. Next the candidate will complete a quantitative multiple choice survey, indicating which practices they used in the project and whether it was *heavily used*, *somewhat used* or *not used*. This is done to make sure the respondents remember all practices. The candidate will then elaborate on these answers in a qualitative open ended style. The goal of which was to figure out *why* they used the practices they used and what the effect of using them were. A final question was added to check if the candidate has anything else he wanted to add. The interview guide is available in Appendix B.

Three types of validity were considered in order to measure its quality: *construct* validity, *internal* validity and *external* validity (Yin, 1994). Validity was achieved by basing the interview guide on a thorough literature review and conferring with colleagues, having multiple sources of evidence (four respondents), and by showing the results to some of the respondents to make sure they were not surprised by them. In order to make sure the study can be repeated with the same results (Yin, 1994), the reliability was also considered. This was achieved by making a case study database with audio files and transcriptions of all the interviews. These can be made available for other researchers.

4. ANALYSIS

In the analysis section, the interview data is presented and analyzed. The analysis is grouped by the project management triangle constraints which were described earlier. Practices described in this section have been identified as *heavily used* by the respondents, and will be analyzed to see how important they are and why. Furthermore, the analysis till focus only on practices which are especially important in agile projects. Practices which are equally important in traditional projects, will not be analyzed.

Out of a total of 53 practices, 23 were found to be heavily used. 15 of these are especially relevant in agile projects. Eight are were related to *scope*, one is related to *time* and six are related to *quality*. The analysis is therefore divided into three groups: scope, time and quality.

--- 1.1 Scope ---

Scope refers to the set of functional elements (features) delivered during the project. Eight practices related to scope were identified as heavily used. *Iterative development, product backlog* and *stories* are heavily used in both projects. *Sprint planning, sprint backlog, sprint review* and *team based* estimation are heavily used in P1. *Incremental design* is heavily used in P2. This is illustrated in **Table 1.**

Scope practices	P1	P2	Both
Iterative development	X	X	X
Product backlog	X	X	X
Stories	X	X	X
Sprint planning	X		
Sprint backlog	X		
Sprint review	X		
Team based estimation	X		
Incremental design		X	
Sum	**7**	**4**	**3**

TABLE 01. Heavily used scope related practices

Iterative development

To be agile, you must be able to quickly adapt to changing priorities. Working in short iterations gives a team this option, by including planning in each development cycle (A. Alliance, 2015). In P1 both respondents ranked this practice with high importance, highlighting *feedback* as the most important reason for doing iterative development. By releasing in small iterations to get early feedback, they can create a product which is more in tune with customer needs.

The developer (D1) mentioned three reasons for doing this practice: solution uncertainty, time to market, and feedback.

"*At the beginning of the project we did not have a 100% clear idea of what we need to do when it comes to features. (…) try to develop (…) in iterations so we can go to market earlier and get feedback from users*" – D1

When asked directly how this practice contributes to success, he (D1) again mentioned stakeholder and user feedback as one of the most important ways to achieve success. The product owner (PO1) had a similar idea of why this is a good practice, and especially mentioned feedback and changing priorities.

"*When things change rapidly it's nice to be able to change direction and not be locked into a predefined course which lasts for 6-12 months. It is also easier to get feedback which benefits the product and the quality. We have chosen two week iterations (…) to get feedback as soon as possible if we do something wrong*." – PO1

The product owner (PO1) also said that it would be difficult to imagine how the project would work without using the iterative development practices. Because they are using Scrum and sprints in P1, it makes it easier to adopt this practice.

None of the respondents in P2 reported this to be a heavily used practice, but they are clearly working in iterations here as well. The main difference is that they do not used fixed time intervals for each cycle, as they do in P1. It therefore seems this practice is heavily used in P2.

"*It is a real cycle (…) it doesn't have a fixed time bound.*" – D2

They work more continuously in P2, taking in prioritized changes when needed. By using a rapid prototyping practice in P2, they can use the iterative approach to quickly respond to changing priorities, get early feedback and gain a quick time to market effect.

"*We do rapid prototyping. There is a designer in the team who (…) comes up with mockups and (…) starts the prototypes (…). The PO uses this prototype to get closer to the customers, to get feedback before starting the implementation. We are starting early and we can also fail early.*"- D2

Even though they have very different approaches on how they do iterative development in P1 and P2, they gain much of the same effects. Working in iterations like this gives the team the ability to create a product which is closer to what the customer wants. It is reasonable to assume that this in turn will increase customer satisfaction, which is likely to affect whether a project is considered successful or not.

Product backlog

Instead of having a large requirement document, agile teams often work from a product backlog which contains the list of features that has to be completed in order for the project to be considered done (A. Alliance, 2015; S. Alliance, 2015). A product backlog is heavily used in both projects but they have a different view on how important it is.

Traditional projects might also break a requirement document into a list of tasks, but for agile project using the iterative approach, it is especially important. In P1 the developer (D1) said that there were two main reasons for using this practice.

"*It sets the overall objective of the product, so we have a long-term mission objective (…) also helps breaking this down (…) and have an idea what needs to be taken into the upcoming sprint.*" – D1

The last point is only important if the team works in sprints. The product owner (PO1) also mentions long and short term planning as important reasons for using this practice. In addition, he said that having a product backlog, if it is prioritized, makes the project more transparent and allows other stakeholders to get an overview of progress and prioritizations.

"*You need a product backlog (…) the product owner side can see what we have planned, but it is also used to create transparence with the team*" – PO1

Both the developer (D1) and the product owner (PO1) made a point of mentioning that this practice is especially important for Scrum, because of the iterative approach. They both ranked this practice with high importance.

Even though they have a product backlog in P2, they do not put a lot of emphasis of the importance of it. Their problem is that priorities are changing very rapidly. Because of this, items on top of the backlog may not get picket for development first. This rapid change made them move away from Scrum and sprints.

This is interpreted to indicate that the product backlog practice is important for some agile methods, but not for all. Methods using sprints or other time boxed iterations, would likely benefit from using this practice.

Stories

Items in a backlog can be structured in a lot of ways, but creating stories is a popular choice. The user centric way of writing stories makes it easier for the developer to understand who the user is and why he needs this story (A. Alliance, 2015). This practice is heavily used in both projects, but the respondents disagree on how important it is. The main incentive for creating stories were reported by all respondents to be the ability for everybody involved to understand why a feature is important for the end user.

"*It is a nice and short way to (…) get a picture of what the customer needs. It's a good way to make all team members quickly understand what the feature is all about.*" – PO1

"*I write user stories so they* (i.e. the developers) *can see what the end result of a task should be, in the eyes of the user. This way they have to understand the user perspective. We feel that this works.*" – PO2

In P1 the developer (D1) also said that this makes it easier to focus on the user's objectives rather than just completing a feature. It affects development in a positive way. He also emphasized that he felt this practice should be used by everybody because "*ultimately the product success depends on whether users get what they expect or not*". The rest of the respondents said that this practice is important to achieve success, but many other practices are more important. Furthermore, it seems like this practice is important regardless of which method is used, because creating stories helps the team to understand why and how the user intends to use a particular feature.

Sprint planning

When a team works in iterations, sprint planning helps to plan each iteration before it starts (S. Alliance, 2015). This practice is heavily used in P1, but this is mainly because they are using Scrum. To be able to commit to a set of tasks each sprint, they need to plan the sprint first. The product owner (PO1) mentioned that the alternative, planning several months ahead, is a bad idea.

"*We plan often and for a few features at a time, instead of having huge planning sessions. It causes us to work only with features which are important right now. We plan the most important things at any given time. It is important.*" – PO1

Both respondents in P1 indicates this as an important because it helps the team to build the right thing. The product owner (PO1) said that it was very important, while the developer (D1) only

found it important because they are using Scrum.

In P2, which has a more continuous approach, they no longer use this practices, at least not in the same way. When they were using Scrum, this practice was heavily used in P2 as well.

They still do planning sessions, but more infrequent and supplemented by regular calls between the PO and individuals in the development team. This infrequent setup works well for them because their process is more continuous and there is no need to plan the next few weeks regularly. Because they have so many changes in priority, planning too far ahead would just be a "waste of time" (PO2).

Sprint backlog

During one of these planning sessions, a sprint backlog is often created and contains all the features selected for development in the upcoming sprint (S. Alliance, 2015). This practice is heavily used in P1, mainly because they are using Scrum.

The product owner (PO1) mentioned that having sprints and iterations works very well. One reason is that P1 is a development project with little or no running business. I.e. bug fixes and other interferences. There are few change requests, and this makes it easier to use a sprint backlog.

"*We have a sprint backlog. It is also prioritized, so we know what should be solved first and last. It's a good aid to get an overview of each sprint.*" – PO1.

The developer (D1) feels the main effect of this practice is to see if the team is succeeding in completing all items in the sprint. The product owner (PO1) said that being able to completely focus only on a few items, and not having to consider the whole backlog when deciding what to do next, gives the team more focus and makes communication easier. He also likes that it gives him some way to measure progress.

They do not use a sprint backlog in P2. This makes sense since they do not work in time- boxed sprints. They used this practice in P2 when they were using Scrum. This worked for them, and at that time they felt it was an important part of the process.

Using a sprint backlog seems important for Scrum projects, but not for agile methods with a continuous approach.

"*In a Scrum setting it is very important. But not equally important in other agile methods.*" – PO1

Sprint backlog might be an important practice, but all respondents in P1 agrees that it is not a critical factor for achieving success.

Sprint review

At the end of every sprint/iteration, a sprint review is often held. This is an informal meeting between stakeholders and the development team in order to get feedback (S. Alliance, 2015). This practice is heavily used in P1. It is a prescribed practice in Scrum, but both respondents in P1 said that this in a very important practice because of the feedback it generates from important stakeholders.

"*The main advantage was to make sure we demonstrate what we have achieved (…) and to get the feedback. Mainly to get feedback.*" – D1

"*Early feedback. To get that feedback loop as short as possible is important. To avoid that the development moves in the wrong direction.*" – PO1

The product owner (PO1) also mentioned that a lot of people feels there are too many ceremonies (i.e. events) in Scrum, but he feels that this team and this project benefits from all of them. He specially mentions the importance of planning each iteration and reviewing the work and the process after each sprint.

They do not have reviews in P2 anymore. At least not in the common sense. Earlier they included everybody in this meeting, but now it is just the PO and one developer. The PO then informs the rest of the company. They do this because they feel it is more effective. This approach might be more effective, but it is not stimulating feedback in the same way. They compensate the lack of feedback from stakeholders by being in constant contact with customers and checking what they think about new features. The feedback arrives later, but it is probably more accurate because it comes directly from the end users.

This practice is considered important because it stimulates early feedback. The interpretation of this is that having a sprint review, or some other way to stimulate feedback is important for success

Team based estimation

To make it easier to know how many items you can expect to complete in an iteration, they can be estimated. In team based estimation this is done by the whole team together. This practice is heavily used in P1 to get a more accurate estimation. A good estimation will in turn make it easier to plan the iteration.

"*The effect is that you get a more realistic estimation (…) we measure velocity based on estimations.*" – PO1

"*It works well, because different team members will have different amount of overview into the code and the complexities involved. We get more accurate estimate for a task.*" – D1

However, it certainly takes more time when everybody in a team have to be present during the estimation process, and this is why they no longer follow this practice in P2. They do not feel that the extra effort of estimating is worth the extra accuracy.

"*We did it earlier at the start of the project. Now (…) it just takes unnecessary time*" – PO2

Team based estimation does not appear to be the most important practices to use to succeed in a project. However, if you need estimations to be more accurate, this is a smart thing to do.

Incremental design

Incremental design is used in order to design the system throughout the lifetime of the project, in order to make sure you have all the necessary information available before making decisions (A. Alliance, 2015). This practice is heavily used only in P2. The developer (D2) indicated this to be a very important practice for achieving success.

"*We think ahead and use whatever the design is for now. We keep it flexible for the future.*" – D2

The PO in P2 strongly feels that they cannot know what the customer needs 100%. So, they plan a feature to about 60%. They then talk with their users and figure out the remaining 40% before they start developing it. This affects the scope of the product and he feels that it makes the product more usable and competitive in the market.

"*We can't imagine what the customer needs 100% in the future. We ask the customer.*"– PO2

In P1 they have made some design choices in the past which now hampers productivity.

"*We don't think years ahead, but some mistakes were done earlier. It was a bit dumb (…) results in a bit slower productivity.*" – PO1

They (P1) have recently started using this practice because it has become evident that designing a solution too far ahead is not a good idea. Priorities change, and features that are important now, might be complete irrelevant in a few months.

Based on the success of using this practice in P2 and the lack of success by not using this practice in P1, the conclusion is that this is indeed an important practice for agile software development and project success.

--- 1.2 Time ---

Time refers to the amount of time required to complete the project with its scope. The *whole team* practice was the only agile practices related to time which was identified as heavily used.

Whole team

The principle of *whole team* is that the team should possess the skills and knowledge they need to complete the assigned work (Beck & Andres, 2004). This practice was embraced in both projects by ensuring that all the needed skills are found in the team.

"*We don't have any designers, but we can get one on request. But the rest of the team should be able to do everything else. That's the principle.*" – PO1

"*We have what we need in the team. When we add new people we make sure they have the competency we require. They should also add something extra to the team.*" – PO2

"*The responsibility goes really high, because you have to find solutions and also make it work. The responsibility and the ability to commit becomes very high.*" – D2

When you empower the team, and trust them to get the job done, it motivates them and increases their responsibility. It stands to reason that this will also reduce the time needed to complete features. The importance of this practice, in terms of project success, is unclear. All respondents had different opinions. The developers found it more important compared to the product owners. Another issue is that this practice might be important in traditional practices as well. The conclusion is that this is not one of the most important practices for achieving projects success in agile software projects.

--- 1.3 Quality ---

Six agile practices designed to increase product quality were identified as heavily used. These practices are all related to process quality which refers to how the system is created and the process around it. *Task boards* and *visualize workflow* are heavily used in both projects. *Daily standup, product owner* and *Sprint retrospective* are heavily used in P1. *Improve collaboratively* is heavily used in P2. This is summarized in **Table 2**.

Quality practices	P1	P2	Both
Task board	X	X	X
Visualize workflow	X	X	X
Daily standup	X		
Product owner	X		
Sprint retrospective	X		
Improve collaboratively		X	
Sum	**5**	**3**	**2**

TABLE 02. Heavily used quality related practices

Task board / visualize workflow

Task board and visualize workflow are very similar practices. The difference is that visualize workflow states that progress should be visible, but not necessarily how it should be done.

Using a task board is one way to visualize progress, by moving tasks from left to right on a board with status columns (A. Alliance, 2015; Hammarberg & Sundén, 2014). These practices were viewed as the same thing by the respondents. However, they did not agree on how important they are. They rely heavily on a digital task board in P1. Having a way to see who is working on what, makes the whole iteration more transparent for everybody.

"*Task board help us visualize the work that we have to do for the particular iteration and where it stands at a particular day or a particular moment.*" – D1

In P2 they have a digital *and* a physical white board. The digital board is available to everyone, while the physical board is only available to the developers. The digital board is not updated as frequently as the physical board.

"*The best thing we are using is the physical white board. Details can be found in the digital board.*" – D2

Most respondents indicated this to be moderately important to achieve project success, but they also said it was a very helpful practice to achieve transparency in a team.

"*It might not be that important for whoever is working on a task (…) more important for others who want an overview of the status of a sprint.*" – PO1

In conclusion, this looks like a moderately important practice for achieving project success.

Daily standup

Daily standup is a practice where the development team meet every day for a few minutes, to coordinate development and share important information (A. Alliance, 2015). They use this practice in P1. They do this to update each other on progress and other relevant information. They consider this an important practice, especially since the PO and the developers are not co-located.

"*Daily standup is especially important since they (*i.e. the developers*) are located in Sri-Lanka. Communication is important in all type of work you do (…) regular communication with those you work with (…) to increase communication in the team. Everybody knows what's going on. This is probably reflected in the quality of the product.*" – PO1

They do not have daily meetings like this in P2 anymore. They used to talk every day when they were using Scrum, but now the whole team meets (i.e. video conference) a couple of times a week. The main reason for not doing it daily anymore, is that the team has matured and are more familiar with the domain. There are less questions and impediments. If a developer has a question for the PO, he just calls him directly. The PO in P2 emphasizes that they had good experience with daily standup in the past, but he now feels it is an unnecessary time consumer.

The importance of this practice is somewhat unclear. It seems that respondents in P1 finds it very important, while respondents in P2 does not. One likely reason for this is that the PO in P1 also works partly as a developer. This is not the case in P2. This is likely to increase the need for synchronization in P1. It could indicate that this practice is very important when developers are not co-located, but it might not be equally important when all developers sit together and only the PO is located elsewhere.

Product owner

Having a product owner (PO) who manages the backlog is a great way to make sure everybody knows what the priorities are. In addition, this practice is further enhanced if the PO is dedicated to this role and/or if there is only one (single) PO in the team (S. Alliance, 2015). A PO exists on both teams. None of them are dedicated as they have other roles in other teams. In P1 they have a single product owner, while it seems they have more than one in P2.

In P1 they agree that having a single PO is important to have someone who has the complete overview of current and future requirements, as well as having only one single person responsible for making priorities.

"*We only have one product owner. It creates clarity and makes communication easier. Generally, I would say that it helps the whole team since they don't have to make too many decisions* (i.e. regarding priorities)." – PO1

In P2 it was a bit unclear who the PO is. The PO said he was the PO. However, the developer (D2) mentioned two persons when he talked about the PO. It might be just one PO in the team, but additional people with PO authority are also involved.

"*I have many roles, but I am the PO of the team. It is my responsibility.*"– PO2

"*We have a new PO* (i.e. not PO2). *He is less available on fixed times, but we communicate more (…) in random time slots.*" – D2

The PO in P2 is less available now compared to when they had daily standups and were doing Scrum. There are fewer fixed meetings, so the developers must contact the PO directly if they have any questions. The developer (D2) said that this is a setup that works. However, it does not sound like they have a single PO. At least there might be some confusion on who has that role.

Having a single PO was considered very important for achieving success by both respondents in P1. In P2 they did not consider it important, and this is likely the reason why they do not use this practice to the same extent.

Sprint retrospective / improve collaboratively

Two additional overlapping practices are sprint retrospective and improve collaboratively. Although slightly different, both practices are about improving the team and the process (S. Alliance, 2015; Hammarberg & Sundén, 2014). The respondents more or less viewed them as the same practice. In P1 they focused on sprint retrospective. In P2 they focused on improve collaboratively. However, they did report the same reasons and benefits from using these practices.

"*It's important to figure out if productivity is lower than it should be. We use retrospective to figure out these things. It is likely the most important thing we do.*" – PO1

"*It's* (i.e. improve collaboratively) *an open discussion where anyone can tell anything regarding what we are doing wrong. What are the pain points? It's very good.*" – D2

It might not be accurate to say that both projects are using the sprint retrospective practice *and* the improve collaboratively practice, but they are without a doubt trying to improve continuously by discussing things that is not working and trying to come up with ways to improve. They all agree that this is an important practice.

If these practices can be considered the same practice, it makes it the only practice where all respondents agree that this is a very important practice to achieve project success. Having a way to constantly improve the team and the process is clearly important and stands out as one of the most important agile practices for achieving success.

5. CONCLUSION

The goal of this paper was to look for agile practices which contributes to success in agile projects. 53 practices were considered and 23 were found to be heavily used. Out of these, 15 were found to be especially relevant to agile projects and considered important by at least one of the two teams in the study. However, these 15 are not considered equally important. **Table 3** lists these practices, ordered by how important they are to achieve project success.

Practices	Importance	Scope	Time	Quality
Iterative development	1	X		
Sprint review	1	X		
Incremental design	1	X		
Sprint retrospective	1			X
Improve collaboratively	1			X
Stories	2	X		
Product backlog	2	X		
Sprint planning	2	X		
Task board	2			X
Visualize workflow	2			X
Daily stand-up	2			X
Product owner	2			X
Sprint backlog	3	X		
Team based estimation	3	X		
Whole team	3		X	
Sum		**8**	**1**	**6**

TABLE 03. Heavily used practices ordered by importance

Five practices were found to be especially important. The most important reason for using *iterative development* and *sprint review* were reported to be early feedback from customers. *Iterative development* was also used to handle changing priorities and solution uncertainty, as well as achieving a shorter time-to-market. The principal reason for using *sprint retrospective* and *improve collaboratively* were to improve the process to improve the team. The goal of which was to achieve a better working environment and a faster working pace. The consensus was that these two are probably the most important practices a team can use. *Incremental design* (as well as *stories* and *sprint planning)* were used to respond to changing priorities and customer needs.

Ten additional practices were found to be important, but less important than those already mentioned. The reasons for using a *product backlog, task board, visualize workflow* and *dedicated and/or single product owner* were: increased transparency, project overview and the ability to plan better. *Daily standup* and *sprint backlog* were mostly used because they improved communication within the team. *Team based estimation* and *whole team* were used to increase estimation accuracy and improve development time respectively.

It is worth mentioning that although some of these practices are reported to have an importance of two in **Table 3**, some might be more important for certain types of agile methods. *Product backlog* and *sprint planning* were reported to be very important by the Scrum team (P1). This indicated that, which practices are considered most important also depend on which agile method the team is using.

All practices considered especially important are related to *scope* and *quality*. This is not surprising since all respondents are considered internal stakeholders and, as described in the Literature review section, internal stakeholders often use scope and quality as the most important criteria for determining success.

In conclusion, of the 15 important agile practices identified, *iterative development, sprint review, incremental design* and *sprint retrospective / improve collaboratively* are considered most important. These results seem to indicate that practices which improves customer feedback and the team process, as well as those helping the team to understand customer needs, are important practices to achieve project success.

LIMITATIONS

The study was conducted with only four participants. Adding more participants would affect the results. In addition, because both projects use or had previously used the scrum framework, they are more likely to be familiar with practices associated with this framework.

ACKNOWLEDGEMENT

We would like to thank everybody who participates in the study for their time, thoughts and insights. ◆

• APPENDIX •

APPENDIX A

--- Agile Practices ---

The agile practices identified in the literature review is listed below. Practices are grouped by the agile method/framework they originated from. Practices are not listed in any specific order.

Scrum

1. Iterative development
2. Product backlog
3. Sprint backlog
4. Sprint retrospective
5. Definition of Done
6. Burn down chart
7. Scrum of Scrums
8. Daily stand-up
9. Sprint planning
10. Sprint review
11. Product owner
12. Scrum master
13. Team member

XP

1. Whole team
2. Sit together
3. Weekly planning
4. Quarterly planning
5. Slack
6. Energized work
7. Real customer involvement
8. Incremental deployment
9. Team continuity
10. Single codebase
11. Daily deployment
12. Test-first programming
13. Incremental design
14. Stories
15. Ten-minute build
16. Continuous integration
17. Shrinking teams
18. Root-cause analysis
19. Shared code
20. Code and test
21. Negotiated scope contract
22. Pay per use
23. Informative workspace
24. Pair programming

Kanban

1. Visualize Workflow
2. Limit WiP
3. Manage flow
4. Improve collaboratively
5. Make process policies explicit

Others

1. Task board
2. Coding standards
3. Refactoring
4. Unit testing
5. Team-based estimation
6. Integration testing
7. Story mapping
8. Test-Driven development
9. Agile games
10. Single team
11. Automated acceptance testing

APPENDIX B ---Interview Guide ---

A.1 Introduction - In part one, background information on 1) the company, 2) the candidate and 3) the project the candidate has in mind during the interview is collected.

Question 1: Company background
- Please gives some background information on the company.
- What type of business is the company? (What is the domain)
- Which products do the company have?
- Is there any written information available? Product sheets etc.

Question 2: Candidate background
- How long has the candidate worked at this company and/or in this domain?
- How many years of agile experience does the candidate have?
- How many years of traditional experience does the candidate have?
- Which roles has the candidate held in traditional projects? (Developer, project manager, executive, etc.)
- Which roles has the candidate held in agile projects?

Question 3: Project background (Think of one specific project) What was the name of the product being created? (Project name)
- What was the nature of the product? (Why was it created)
- When was this project running? Is it still running?
- What was the candidate's role in this project?
- Which other roles were defined (PO, SM, PM, Developers, QA, UX, etc.)?
- How many team members were there in this project?
- Who was the target audience?
- Was is considered a success?
- Anything else to say about the product/project?

A.2 Practices - In part two, information about the practices used in the project is collected.

Question 4: Open-ended discussion about the practices used in the project.
- The candidate gives an accurate account of how a cycle/iteration/sprint in the project was conducted. Focus on processes and agile practices.

Question 5: Quantitative survey on agile practices.
- For each practice in the multiple-choice survey (Table 4) indicate if it was used in this project. (Heavily used, somewhat used, not used, don't know)

Question 6: Qualitative open-ended discussion based on answers given in question 5. For each of the heavily used practices, why were they used and what was the effect (positive/negative) of using them?
- For each of the somewhat used practices, why were they only somewhat used?
- For each of the not used practices, why were they not used?

• AUTHORS •

ANDRÉ HENRIKSEN has a Master's degree in Computer Science and a Master's degree in Business Administration. Both from UiT The Arctic University of Norway. He has worked for more than 10 years as a software developer and project owner in various agile projects. He is currently enrolled in a PhD program at UiT The Arctic University of Tromsø. His fields of interest include software development, agile projects management and medical informatics.

SVEN ARNE R. PEDERSEN is an Assistant Professor at UiT The Arctic University of Norway. His fields of interests include project management, entrepreneurship, business strategy, marketing, organizational theory and management. He currently teaches project management.

• REFERENCES •

Agarwal, N., & Rathod, U. (2006). Defining 'success' for software projects: An exploratory revelation. International Journal of Project Management, 24(4), 358-370. doi:http://dx.doi.org/10.1016/j.ijproman.2005.11.009

Agile Alliance. Agile Manifesto. Retrieved from http://agilemanifesto.com

Alliance, A. (2015). Guide to agile practices. Retrieved from http://guide.agilealliance.org

Alliance, S. (2015). Scrum Guide. Retrieved from https://www.scrumalliance.org/why-scrum/scrum-guide

Atkinson, R. (1999). Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. International Journal of Project Management, 17(6), 337-342. doi:http://dx.doi.org/10.1016/S0263-7863(98)00069-6

Beck, K., & Andres, C. (2004). Extreme Programming Explained: Embrace Change. Reading: Addison-Wesley Professional.

Boynton, A. C., & Zmud, R. W. (1984). An assessment of critical success factors. Sloan Management Review (pre-1986), 25(4), 17-27.

Bryde, D. J., & Robinson, L. (2005). Client versus contractor perspectives on project success criteria. International Journal of Project Management, 23(8), 622-629. doi:http://dx.doi.org/10.1016/j.ijproman.2005.05.003

Chappell, D. (2013). The Three Aspects of Software Quality: Functional, Structural, and Process. Retrieved from

Charette, R. (2005). Why software fails. IEEE Spectrum, 42, 42-49.

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

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. Information and Software Technology, 50(9–10), 833-859. doi:http://dx.doi.org/10.1016/j.infsof.2008.01.006

Griffiths, M. (2012). PMI-ACP Exam Prep: RMC Publications, Inc.

Hammarberg, M., & Sundén, J. (2014). Kanban in Action: Manning Publications Co.

Jalali, S., & Wohlin, C. (2012). Global software engineering and agile practices: a systematic review. Journal of Software: Evolution and Process, 24(6), 643-659. doi:10.1002/smr.561

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., . . . Zelkowitz, M. (2002). Empirical Findings in Agile Methods. In D. Wells & L. Williams (Eds.), Extreme Programming and Agile Methods — XP/Agile Universe 2002 (Vol. 2418, pp. 197-207): Springer Berlin Heidelberg.

Mar, K., & Schwaber, K. (2002, March 22, 2002). Scrum with XP. informit.com.

Mens, T. (2008). Introduction and roadmap: History and challenges of software evolution: Springer.

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

Project Management Institute. (2013). A guide to the project management body of knowledge: (PMBOK guide). Newtown Square, Pa.: Project Management Institute.

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

Siddique, L., & Hussein, B. A. (2016). A qualitative study of success criteria in Norwegian agile software projects from suppliers' perspective. IJISPM-INTERNATIONAL JOURNAL OF INFORMATION SYSTEMS AND PROJECT MANAGEMENT, 4(2), 65- 79.

The standish Group. (2015). CHAOS Report. Retrieved from

VersionOne. (2015). State of Agile Survey web page. Retrieved from http://stateofagile.versionone.com

VersionOne. (2016). 10th Annual State of Agile survey. Retrieved from

Wicks, A. M., & Roethlein, C. J. (2009). A Satisfaction-Based Definition of Quality. The Journal of Business and Economic Studies, 15(1), 82-97,110-111.

Williams, L., Brown, G., Meltzer, A., & Nagappan, N. (2011, 22-23 Sept. 2011). Scrum + Engineering Practices: Experiences of Three Microsoft Teams. Paper presented at the Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on.

Wysocki, R. K. (2014). Effective project management: traditional, agile, extreme. Indianapolis, Indiana: Wiley.

Yin, R. K. (1994). Case study research: design and methods (2nd ed.). Thousand Oaks, Calif.: Sage.

Yin, R. K. (2014). Case study research: design and methods (5th ed.). Thousand Oaks, Calif.: Sage.