Scientific Research Publishing

American Journal of Industrial and Business Management, 2023, 13, 183-194
https://www.scirp.org/journal/ajibm
ISSN Online: 2164-5175
ISSN Print: 2164-5167

The Impact of Agile Methods on the Software Projects Implementation and Management

Nelsen A. Rahul, Thierry S. Nouidui, Paul L. Ulaya, David D. Kiwia

United African University of Tanzania, Dar es Salaam, Tanzania
Email: drnelsen@uaut.ac.tz

How to cite this paper: Rahul, N. A., Nouidui, T. S., Ulaya, P. L., & Kiwia, D. D. (2023). The Impact of Agile Methods on the Software Projects Implementation and Management. *American Journal of Industrial and Business Management, 13,* 183-194.
https://doi.org/10.4236/ajibm.2023.134013

Received: March 8, 2023
Accepted: April 14, 2023
Published: April 17, 2023

Abstract

There are emerging patterns of failure and success as more software projects adopt Agile methods. Project managers now need to understand the factors that influence project performance, as well as the applicable practices to their projects. Although some companies claim that Agile methods can solve all their issues, few have consistently shown success over other types of software projects. Although Agile methods can accommodate change due to the volatile requirements, they can also expose project managers to risks when it comes to managing the multitude of pieces of work involved in a project. This paper aims to provide a comprehensive analysis of the various trade-offs that are involved in adopting the method.

Keywords

Agile Method, Software, Extreme Programming, Scrum, DSDM Methods

1. Introduction

As a discipline, software engineering faces two main challenges. One of these is the complexity of software, which is often more complex than steel or integrated circuits. Because of this, the cost of implementing changes during a project's lifecycle is higher (Highsmith & Cockburn, 2001). Since design and requirements can be easily tested, the use of Agile methods has become more prevalent. These methods help organizations manage the risks associated with software development (Beck. *eXtreme Programming Explained.* Addison-Wesley, 2000). Some of these include the SCRUM (http://www.controlchaos.com/about), eXtreme Programming (Beck. *eXtreme Programming Explained.* Addison-Wesley, 2000), and DSDM (Stapleton. *DSDM—Dynamic System Development Method.* Addison-Wesley, 1995) methods. Although the exact details of these methods vary, they all share a common goal of helping teams respond faster to changes. Since it is

costly to accommodate changes later in the project (Paetsch, Eberlein, & Maurer, 2003; Thomas, 2022), adopting agile methods can help reduce the risk of project delays and costs (Beck. *eXtreme Programming Explained*. Addison-Wesley, 2000). However, in some cases, it is not feasible to implement these methods due to the complexity of the software. When it comes to adopting an agile method, managers have to decide if it is appropriate for their project. They should also consider the various risks associated with this approach. Understanding these risks and managing them is very important for software project management.

In this paper, we discuss the various advantages and disadvantages of adopting agile methods for software project management. We also talk about their impact on the people, process, and project (Phillips, 1998).

2. A Brief Look at Agile Methods

Over the past decade, the number of Agile Methods has increased significantly. This shows that the principles they preach warrant further examination. We present a brief overview of the various frameworks that are used in the practice, and then we discuss their convergence in the Manifesto for Agile Software Development (http://www.agilemanifesto.org).

2.1. Extreme Programming

The concept of extreme programming, also known as XP, is a method that focuses on getting the project done. It doesn't involve using any magic bullets or fancy techniques. Instead, it uses a series of principles to get the job done. The life cycle of this method is composed of five phases.

During the Exploration phase, the customers provide the necessary details for the first release. The project team then gets familiar with the various tools and practices that will be used during the development of the project. In the planning phase, the team focuses on identifying the capabilities that will be needed to launch the first version. The team leader then draws up a release schedule that is expected to last for two months. During the iterations to release phase, the team members perform various tests to determine the features that will be included in the first release (Beck. *eXtreme Programming Explained*. Addison-Wesley, 2000).

The last iteration of the project is considered finished and ready for production. During the production phase, the team carries out additional performance tests and checks to ensure that the release meets the customers' requirements. After this, the team decides if new features should be added to the current version. If the new features are not included in the current version, they will be archived for subsequent releases. The release is delivered to the customer at the end of this phase. During the Maintenance phase, which is the next step, the team works on implementing the changes that were raised during the previous stage. The various changes that were made during the maintenance phase are usually categorized into corrective, adaptive, and perfective changes. As the software

gets older, and customers have fewer features that they can implement, the Death phase involves completing the documentation and decommissioning the system. This step is usually triggered by the lack of a compelling value proposition for the system.

2.2. SCRUM

The SCRUM (http://www.controlchaos.com/about) development process involves various technical and environmental variables that can change during the course of a project. This process is designed to help teams keep their focus on developing software while constantly adapting to the changing environment. The three phases of SCRUM are the pre-game, development, and post-game. The pre-game phase is divided into two parts: planning and architecture/high-level design. The planning stage involves gathering information about the system's requirements and developing a list of features and modifications.

The architecture sub-phase focuses on the refinement and evolution of the design based on the backlog list. During the development phase, the team goes through iterative cycles to enhance the system and add new functions. Each sprint involves a variety of tasks such as analysis, design, evolution, and delivery.

Each sprint is typically executed in one-to-one month. During the development stage, around three to eight sprints are carried out before the system is ready to be released. The post-game period, which is the end of the effort, provides the system with no additional modifications or features.

2.3. Dynamic System Development Method

DSDM utilizes a method that focuses on the optimization of resources and time while also adjusting the amount of functionality that can be added to the system. This process is divided into five phases. The last three phases of the process are incremental and iterative. These are designed to restrict the number of iterations that can be carried out within a given time-box. During the feasibility study phase, the team decides if DSDM is the right method for the project.

A feasibility report and development plan are then produced. The business study phase focuses on the key characteristics of the project and the technology. This stage also involves developing an outline prototyping plan and a system architecture. The development phase begins with a prototyping plan that describes the strategy and the approach to configuration management. The functional model iteration stage involves carrying out functional iterations, which are usually focused on improving the system. The four products that follow this phase reflect the process's various steps. These include reviewing documents related to the project's requirements, identifying potential risks, and prioritizing the tasks.

The build and design iteration is a process that involves validating the requirements of the customer and then coming up with a solution that meets those requirements. Through a series of iterations, the software is refined and im-

proved in a consumable form for review.

The system is then transferred to the customer during the implementation phase, and subsequent upgrades are planned.

2.4. Agile Manifesto Ties It Together

The goal of the Agile Manifesto is to provide a comprehensive overview of the principles that are used in developing software (http://www.agilemanifesto.org).

- ☐ individuals and interactions over process and tools
- ☐ working code over comprehensive documentation
- ☐ customer collaboration over contract negotiation
- ☐ responding to change over following a plan

The Manifesto acknowledges that the second item is not as important as the first, but it is still less important than the first. Software constantly changes, which makes it incredibly challenging to keep up with. This is why it is important that people take on the slack. People over process can lead to better solutions. It is also implied that even the best processes cannot compensate for people's shortcomings (Cockburn & Highsmith, 2001). While documentation is valuable, it can take a long time to maintain and write. Some methods, such as Agile Methods, encourage the rapid prototyping of products, while others, like XP, prefer to build simple, yet functional solutions.

In agile methods, customer involvement is promoted. A knowledgeable and dedicated representative is needed to be available to answer questions and provide insight. This form of collaboration allows customers to change their minds. Instead of writing contracts, customers should be involved in the development process. They should also consider responding to changes as important as following a plan.

The plan must also be updated as changes happen. This is because, in agile methods, the goal is to create a plan that is easy to modify and lightweight. This does not mean that the method assumes a hacking mentality. Instead, it emphasizes the importance of having a plan that is easily modifiable. The "plan" might be a set of notes that are made on a whiteboard, similar to what SCRUM uses. This paper aims to support the four values of agile methods by examining their similarities. Instead of going through the various nuances and differences of each method, this paper focuses on the project manager's perspective (Derbier, 2003).

The six main features of agile methods are collaboration, code reviews, small teams, short release schedules, time-boxing and constant testing. All of these are highly collaborative methods, which are very different from traditional methods. In agile methods, communication is informal, and information is quickly spread to other people. Any method that doesn't foster a collaborative environment is doomed to failure. The project manager is responsible for ensuring that the environment is conducive to a productive collaboration. In addition, agile methods encourage code reviews.

Code reviews allow for the dissemination of important information, such as

technical details. For instance, in XP, continuous code reviews are carried out through pair programming, where two developers share one computer. In agile methods and DSDM, small teams are also encouraged (http://www.balagan.org.uk/work/agile_comparison.htm). Typically, there are between three and six teams working on a project. The goal of small teams is to foster collaboration, which is more likely to reduce the amount of planning and process involved in the project. With agile methods, release schedules are typically shorter than two weeks. They also allow for the continuous evaluation of the product and the addition of new features (Boehm & Turner, 2003).

Unlike traditional methods, which focus on the fixed features and a fast delivery date, time boxing allows for the continuous development of the product. This method helps reduce the risk of scope creep and gold-plating while also focusing on the customer. To ensure that the product quality is maintained, agile methods typically test the product continuously throughout its lifecycle. The goal of test-first is to offset the risks associated with just writing the code. In agile methods, continuous testing is carried out throughout the entire development process. This method requires the use of automated tests to ensure that the product works seamlessly (http://www.balagan.org.uk/work/agile_comparison.htm).

3. Impact on Project Management

Although Agile methods are generally good ideas, they can also have significant impacts on the people and processes involved in a project. This paper aims to identify the various impacts that can be considered when it comes to adopting an Agile method for a project.

3.1. People

Developers, testers, and project leaders are some of the individuals who can be involved in a software development project. There are also various other individuals who are interested in the project's success, such as business managers and directors of development shops.

3.1.1. Developers

One of the biggest impacts of Agile methods is on the developers. The individuals who are involved in the project must be skilled and talented. They should also be able to communicate well and work as a team.

Since Agile methods are very lightweight, they do not provide the necessary guidelines and processes that developers need to follow. This can make them unsuitable for weaker developers.

The "−1" level of developer depicted in Table 1 would be challenged in an agile environment. Even "1B" developers consume resources in "hand-holding". Hence, the top three levels make up the core of the agile development team. Boehm and Turner suggest level "3" developers may not be needed for all projects,

Table 1. Boehm & Turner's developer levels.

Level	Characteristics
3	able to produce solutions in unprecedented situations
2	able to tailor solutions to fit new, but precedented situation
1A	solid developer able to implement functionality, estimate effort, & refactor code
1B	able to implement simple functionality, execute tests, & follow directions
−1	unwilling or unable to work in a collaborative environment

depending on how unprecedented it might be. Given the need for a high level of expertise, Agile Methods may be difficult to employ in a traditionally staffed organization. Highly skilled staff are always in demand, and without accommodating 1B developers, it may be difficult to build a long-term human capital strategy. This is just one reason that long term projects present a significant risk for Agile Methods.

3.1.2. Testers

The importance of using an Agile method in testing is that it allows the organization to focus on the development of the code instead of the testing of the entire process. This method tends to reduce the number of testers involved in the testing process. As testers, they typically focus on functional and system tests. They may need to be more capable of handling the automation of these tests, as well as integrating them into the testing framework. This may be a different skill set. The project management team usually has to find new testers who don't fit into the Agile group. This can be done by identifying individuals with the necessary skills and knowledge to handle the various aspects of the testing process. For instance, a novice developer may start with a level 1B certification and gain more Agile Method expertise.

3.1.3. Project Leaders

The two main roles of software development managers are team leads and project managers. They both have their own set of challenges as the management style of Agile differs from other methods. Having a coach or mentor can help the team members manage their own projects and ensure that they are getting the most out of their work. The team lead is responsible for encouraging the members to take ownership of their projects. This type of leadership can be a cultural shift as it requires the team members to share authority over making decisions. Unlike in agile methods, project managers are responsible for making business decisions and tracking progress. This makes the adjustment more significant.

Instead of planning and schedules, agile methods emphasize the ability to respond to changes. This makes it challenging for project managers as they are often called upon to provide detailed information about the projects.

In SCRUM, for instance, the project manager is responsible for leading the daily meetings and interacting with the team. In agile methods, the team lead is

also frequently involved in the customer collaboration. If the project manager is not capable of handling the role of project manager in an agile method, then selecting this method may not be the right choice.

3.1.4. Customers

Instead of having customers only involved in the development of a project, Agile Methods involve them in the project's inception and throughout its lifecycle. This allows them to participate in the development of the project and provide feedback on the quality of the work. In Agile Methods, customers are more involved and have more influence. Most methods recommend having a full-time presence on site, and finding someone who would be willing to do this can be challenging. Some companies find that their customers are unwilling to work with them, while startups might not be able to identify their ideal customers. Having a sufficient number of available customer representatives is also important when using Agile Methods.

The ideal candidate should be knowledgeable, committed, capable, and collaborative. They should also have the authority to make critical decisions regarding the features that will be included in the next release. This type of representative should not be used in every project, as they might not be available for all.

3.1.5. Executive Management

One of the most critical factors that an organization should consider when selecting a new process is the availability of executive management support. This is especially important since, in Agile Methods, the executive managers are usually risk-averse. They also want to see definite delivery dates and progress on various tasks. The cultural change that Agile Methods brings is very different from what they used to experience. In addition to having fewer documents to track progress, it also allows features to change rapidly as the process continues. One of the biggest issues that they encounter is the lack of a priori knowledge regarding the scope and cost of a project.

This situation is not ideal for management, as it can prevent them from adopting an Agile Method. Project managers must convince executives that this method will deliver better quality and faster. If executives are willing to try out the method, the success of the projects that they have started using will determine the method's continued usage. For instance, if the project managers are able to build a strong relationship with their executive management, the results of the project can be very beneficial.

3.1.6. The Team

The team is very important to the success of an Agile project, as it relies on communication and collaboration. If a single developer or a customer does not work well with the team, the project can be severely affected. Another critical personnel issue that an Agile team should consider is turnover. High turnover can lead to the loss of knowledge, especially if there is no formal documentation. Although code reviews and rotating developers around the project can help mi-

tigate this issue, losing a significant member of the team can be very detrimental.

This issue should be considered by the project manager when assessing the viability of an Agile Method for their organization. By retaining skilled individuals, they can ensure that they have the necessary knowledge to carry out their tasks.

3.2. Process

The various processes that an organization uses are affected by the introduction of Agile Methods (Cohn & Ford, 2003). For instance, they have to replace old processes such as planning and development with new ones that are more agile. The cultural changes caused by the method can also cause resistance.

3.2.1. Planning

Instead of placing emphasis on planning, agile processes approach planning as a continuous task that can be performed to ensure that the outcome is optimal. This method is different from other methodologies as it does not involve micro adjustments. In agile planning, the various planning stages are typically handled informally. For instance, the daily SCRUM meeting helps determine what will be included in each time-box. Other methods might not consider this level of planning.

3.2.2. Documentation

The documentation of agile methods is typically sparse, and it is often limited to a few user stories and source code. Most methods allow for the development of an optional architecture, and in some cases, it is mandatory. The documentation is also driven by how often it needs to be updated. One of the most important factors that an agile method should consider when it comes to developing documentation is ensuring that the vision statement is always recorded. This ensures that the project is always on track and that the changes are reflected in the documents.

Information is kept informally within an agile process, and it is shared with the entire organization. Although reducing the number of documents can improve productivity, it can also lead to some cost and risk. Documentation serves as an opportunity to introduce new members to the method. It is also beneficial when transitioning a project from a technical team to a maintenance one. Having the proper documentation helps ensure that the procedures are followed properly, and it can help prevent potential issues from happening.

3.2.3. Development Processes

One of the main principles of agile processes is that they encourage principles that can dramatically change a process. Some of these include continuous integration, refactoring, and code reviews. One of the most common processes that can be performed in an agile process is refactoring. This process involves taking code and improving its functionality without losing its readability. Refactoring involves following defined contracts and tests and ensuring that the code passes

all of them.

The question that comes to mind when it comes to implementing new features is when refactoring is prioritized over adding new ones. In the agile method community, the term "YAGNI" is used to describe a minimalist approach to development. This method eliminates features that are not needed in order to make the implementation simple (Boehm, 2002; DeMarco & Boehm, 2002). The goal of this approach is to reduce the effort involved in the development process while also avoiding unnecessary features. However, it is important to note that this method can lead to less effort later on as the requirements for the system are known. A code review is a process that involves one or more developers looking at the code that another has written. It can be performed as a continuous process or as a periodic procedure, similar to the way in which peer reviews are conducted in DSDM. Code reviews are also beneficial when communicating with one another.

Through code reviews, developers can get a better understanding of the inner workings of the system and the design tradeoffs that they need to make in order to work on the features that they later will need. This can help minimize the risk of having a team member go on vacation or leave due to an employment change. Continuous integration is also beneficial as it allows the team to test the system regularly. In continuous integration, a developer adds their code into a baseline, then tests it against regression tests. This method can increase the quality of the work that they do as it allows them to quickly identify potential issues. One of the most important factors that can affect the schedules of agile processes is the discovery of defects early.

Since developers have to create comprehensive tests to be used as Regression tests, they often have to take a lot of time to integrate their code and test it. This can be a shift in their perspective depending on how they are used to writing code. Most development groups follow these principles. Most developers are not fond of the idea of peer programming, and they may find it hard to justify the large number of tests that they have to perform. The project manager might have to consider incorporating these processes slowly in order to gain their approval.

3.3. Project

Despite the number of proponents of Agile Methods, it is not always clear that this method is applicable to every project. There are various factors that can prevent organizations from adopting this method, such as business factors.

3.3.1. Project Types

Most of the time, agile methods are applicable to projects that have unpredictable requirements. They can be used to accommodate changes easily. Projects that are new within an organization or are heavily impacted by technology are examples of such projects. However, agile development doesn't allow for the type of rigorous analysis that is needed to ensure that the safety of critical systems is protected. This is because it requires a lot of analysis and documentation. While

having a large number of tests can help keep the quality of work, it is only as good as the quality of the tests itself. Although code reviews are typically performed in agile development, they do not follow the same formal methods that are used for critical systems analysis.

3.3.2. Business Factors

One of the most common business factors that can prevent an organization from adopting agile development is the contractual obligation. This is because many companies have a set of requirements that are defined in a statement of work. If the requirements are part of a legal contract and are not addressed in agile development, then this method might not be appropriate. In a contracting relationship, documentation is used to show the status of the project and provide a transition to the company that will be handling the work. In government contracts, there are numerous requirements that are designed to ensure that the documents are in compliance with international standards such as ISO9000.

If a company wants to set release dates for its products, this type of contract is often used. It is necessary for them to know in advance when a new feature will be released so they can plan on how to migrate to it. For instance, if a company uses financial applications or ERP software, they might have to know in advance when a new release will be introduced. If a company's products need a well-defined road map for their features, then agile development might not be the best option. Aside from this, documentation also has regulatory reasons. For instance, in the financial services industry, the SEC requires companies to provide documentation that explains how they can prevent certain issues from happening. Although agile development can be used as a method for creating documentation, it is still important to consider the changes that will affect the documents.

3.3.3. Other Project Characteristics

One of the most important factors that can prevent an organization from adopting agile development is the time span of the project. This is because long-running projects can lead to a lot of staff turnover. One of the most common issues that agile development can encounter is the lack of team knowledge. This can be addressed by implementing a rotation of team members into different areas. Another issue with long-running projects is that they tend to be larger and have a lot of features.

This can lead to issues in prioritization. A single project manager may not be able to handle all the tasks and may have to make decisions based on the priorities of the team. Long-running projects can also have issues with maintenance. In agile development, the lack of documentation can be a major issue. It's also possible that the developers who are responsible for a long-running project have already moved on and might not remember the decisions made in the past. Having a sufficient amount of documentation is necessary for supporting a product that's expected to last for a long time (Boehm, 2002).

One of the most important factors that a long-running project should consid-

er is its project roadmap. This can help the team identify the changes that will affect the product and the requirements of the future. Having a well-defined roadmap can also help a long-running project avoid issues related to the lack of resources and the time constraints of the project.

4. Conclusion

Agile methods are generally considered to be a good alternative to traditional methods due to their ability to handle the high degree of uncertainty and change in today's software development environment. They can also help reduce the risk of a project and improve the quality of work by implementing proven principles. When combined with other agile techniques, these can help increase the success of a project by reducing the risk of errors and improving the quality of work. One of the most important advantages of agile development is its ability to address the changes that affect a project.

Although agile methods can be beneficial for a project, they shouldn't be used for every project. Before implementing them, a project manager should first consider the various characteristics of a project. For instance, if a large team of junior developers is being used for a project with a mature process, there are multiple characteristics that prevent the use of agile methods (Paetsch, Eberlein, & Maurer, 2003; Abrahamsson, Warsta, Siponen, & Ronkainen, 2003).

The principle of small teams is also beneficial for a project. However, it's important to consider the various challenges that agile methods can bring. For instance, if the project is being migrated to a new maintenance group, the development team might have to create documentation. For project managers who are looking for an alternative to traditional methods, agile methods can be a good alternative. They can help them avoid issues related to the lack of clarity and fast-changing requirements of a project. Even if the entire method is not ideal for a particular project, the underlying principles can still be used. Before considering the use of agile methods for a particular project, a project manager should first make sure that the team can handle the requirements of the project.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003). New Directions on Agile Methods: A Comparative Analysis. In *Proceedings of the 25th International Conference on Software Engineering* (pp. 244-254). Institute of Electrical and Electronics Engineers. https://doi.org/10.1109/ICSE.2003.1201204

Beck, K. (2000). *Extreme Programming Explained*. Addison-Wesley. https://doi.org/10.1109/TOOLS.1999.779100

Boehm, B. (2002). Get Ready for Agile Methods, with Care. *Computer, 35,* 64-69. https://doi.org/10.1109/2.976920

Boehm, B., & Turner, R. (2003). Using Risk to Balance Agile and Plan-Driven Methods. *IEEE Computer, 36,* 57-66. https://doi.org/10.1109/MC.2003.1204376

Cockburn, A., & Highsmith, J. (2001). Agile Software Development: The People Factor. *Computer, 34,* 131-133. https://doi.org/10.1109/2.963450

Cohn, M., & Ford, D. (2003). Introducing an Agile Process to an Organization. *Computer, 36,* 74-78. https://doi.org/10.1109/MC.2003.1204378

DeMarco, T., & Boehm, B. (2002). The Agile Methods Fray. *Computer, 35,* 90-92. https://doi.org/10.1109/MC.2002.1009175

Derbier, G. (2003). Agile Development in the Old Economy. In *Proceedings of the Agile Development Conference* (pp. 125-131). Institute of Electrical and Electronics Engineers. https://doi.org/10.1109/ADC.2003.1231462

Highsmith, J., & Cockburn, A. (2001). Agile Software Development: The Business of Innovation. *Computer, 34,* 3. https://doi.org/10.1109/2.947100

Manifesto for Agile Software Development. http://www.agilemanifesto.org

Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements Engineering and Agile Software Development. In *Proceedings of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (pp. 308-313). Institute of Electrical and Electronics Engineers.

Phillips, D. (1998). *The Software Project Manager's Handbook: Principles That Work at Work*. IEEE Computer Society Press.

SCRUM (2001, September). *It's about Common Sense* (pp. 120-122). http://www.controlchaos.com/about

Stapleton, J. (1995). *DSDM—Dynamic System Development Method*. Addison-Wesley.

Thomas, S. (2022). An Agile Comparison. http://www.balagan.org.uk/work/agile_comparison.htm