Review Article

iMedPub Journals
http://www.imedpub.com

American Journal of Computer Science and Engineering Survey
ISSN 2476-2105

2021
Vol. 9 No.5:31

Understanding Agile Project Management

Wumi Ajayi*, P. E. Elias, R. A. Akinyemi, A. O. Lawrence and M. O. Taiwo

Department of Computer Science, University of Lead City, Ibadan, Oyo State, Nigeria

*Corresponding author: Wumi Ajayi, Department of Computer Science, University of Lead City, Ibadan, Oyo State, Nigeria

Wumiajayi1@yahoo.com

Abstract

To create successful software, software teams are required. Most software businesses use agile software development to create high-quality software in less time and at a lower cost. The contribution of successful software initiatives is influenced by several factors. Software team productivity is another essential idea in agile software development that contributes to project success. In an agile software development process, teamwork productivity impacts total project success; consequently, studying team member productivity is of great interest.

Keywords: Software businesses; Programming; Testing; Monitoring

Received: August 10, 2021; **Accepted:** August 24, 2021; **Published:** August 31, 2021

Citation: Ajayi W (2021) Understanding Agile Project Management. Am J Compt Sci Eng Surv Vol.9 No.5:31.

Introduction

Agile team members should be taught to understand and manage productivity factors on a regular basis because they are self-managed. The primary aim, according to the agile manifesto, is to satisfy the client by delivering valuable software early and frequently. Working software should be delivered between a couple of weeks to a couple of months, which thus demonstrates a project's progress. The agile manifesto defined the criteria of software productivity and customer satisfaction by short iterations and deliverables. As a result, examining agile team productivity is critical for success in agile software development; nevertheless, there are various other elements at play, and their impact on agile teams must still be extensively and efficiently investigated. As a result, examining agile team productivity is critical for success in agile software development; nevertheless, there are various other elements at play, and their impact on agile teams must still be extensively and efficiently investigated.

- The art of project management
- Project management phases

Literature Review

The art of project management

Project management, as defined by some authors, is "the application of knowledge, skills, tools, and techniques to project activities in order to meet project requirements"1. As an integral part of software engineering processes, project management, along with business analysis and requirement specification, design, programming, and testing, has been a source of considerable debate for years. According to survey results from the Project Management Institute (PMI), even as business project management procedures evolve, only approximately half of them (54 percent) are fully aware of the necessity and usefulness of these practices. Project management has proven to be a critical component of a company's efficiency and eventual success, regardless of industry. Firms that use proven project management strategies squander 28 percent less money and have 2.5 times more successful projects. According to project management experts, a successful project is one that is not only completed on time and under budget, but also provides the anticipated advantages [1].

Project management phases: Any project, regardless of its scope, should follow a set of steps that can be controlled and managed. A typical project management process, according to the Project Management Institute, contains the following phases:

- Initiation
- Planning
- Execution
- Performance/Monitoring
- The project has come to an end.

These phases describe the project management lifecycle and serve as a roadmap for completing certain tasks. However, this structure is far too broad. Within each phase of a project, there are generally several internal stages. Depending on the scope of work, the team, the industry, and the project itself, they can vary significantly. Humanity has produced a large variety of project management approaches and methodologies to discover a universal approach to project management [2].

Traditional project management methodologies

Traditional techniques follow the above-described classic structure and adopt a step-by-step approach to project execution. As a result, the project proceeds in stages from start to finish, including planning, implementation, and monitoring. This strategy, which is also known as linear, consists of a series of internal phases that are consecutive and carried out in a chronological order. Traditional project management, which is most typically used in the construction or manufacturing industries when little or no adjustments are necessary at every stage, has also found applicability in software engineering. The waterfall model, often known as the waterfall process, has been a popular software development paradigm since the early 1970s.

Waterfall model: The waterfall paradigm places a heavy emphasis on planning and specification development, which can account for up to 40% of the project's time and budget. A precise sequencing of project phases is another fundamental principle of this approach. A new stage of a project does not begin until the preceding one has been completed. The strategy works effectively for projects with a clear scope, a single deliverable, and a set timeframe. The waterfall method necessitates meticulous planning, substantial project documentation, and close monitoring of the development process. This should, in principle, produce in on-time, on-budget delivery, reduced project risks, and predictable final outcomes. However, due to the multiple constraints, the waterfall method is sluggish, costly, and inflexible when applied to the actual software engineering process. In many circumstances, the company's inability to adapt the product to changing market demands leads to a massive waste of resources and eventual project failure **(Figure 1)**.

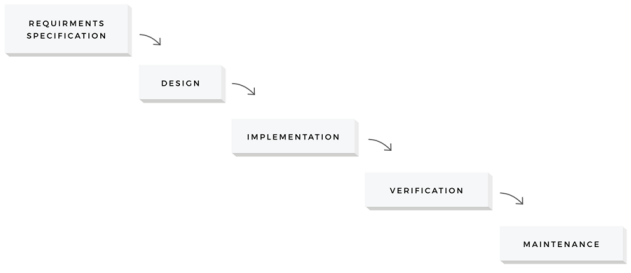

Figure 1 Agile developmental cycle: Waterfall model.

Why do we need agile methodology?

We had the Waterfall model of software development before Agile came around. The waterfall model is a top-down approach to the creation of a system or software that follows a sequential process. This was an easy-to-understand and linear model. Requirements gathering, software design, implementation, testing, deployment, and maintenance were all aspects of the waterfall methodology.

This methodology, however, had a few flaws, including the following:

• It was too time-consuming. You cannot move on to the next step unless you finish the previous one.

• This methodology was only appropriate for projects with consistent requirements.

• The working software is supplied only after the model's final stage has been completed.

• It is tough to go back to earlier phases and make modifications that you hadn't considered during the initial phase.

Agile project management methodology

In contrast to established techniques, the agile methodology was developed to make software engineering more flexible and efficient. Agile project management has become a standard in project management, with 94 percent of firms using it in 2015. The history of agile may be traced back to 1957, when IBM and Motorola were developing software using incremental development approaches (now known as Agile). Despite not knowing how to categorize the technique they were using, they all agreed that it was distinct from the Waterfall in several ways [3]. However, when a group of 17 software development professionals met in 2001 to examine alternate project management approaches, the modern-day agile methodology was officially established. They set out the flexible, lightweight, and team-oriented software development strategy in the Manifesto for Agile Software Development, which they had a clear vision of. The Manifesto clearly explains the core ideas of the new approach, with the goal of "discovering better ways of producing software." The philosophy has become a ubiquitous and efficient new way to manage projects, especially when combined with the Twelve Principles of Agile Software. When it comes to software development, agile techniques adopt an iterative approach. Agile projects, unlike a traditional linear waterfall model, are made up of a series of smaller cycles called sprints. Each one is a mini project with a backlog and stages for design, development, testing, and deployment within a pre-determined scope of work [4].

Agile development cycle: A potentially shippable product increment is given at the end of each Sprint. As a result, new features are added to the product with each iteration, resulting in progressive project expansion. The risks of producing a possibly failing product are considerably reduced when features are evaluated early in the development process **(Figure 2)**.

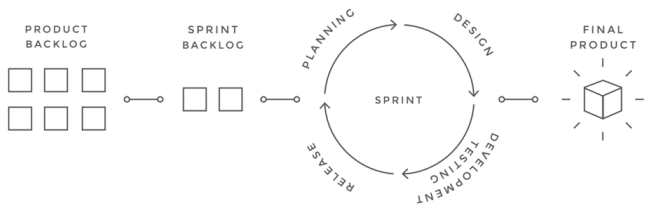

Figure 2 Agile developmental cycle for obtaining final product.

The main agile aspects

Flexibility: As new requirements arise; the scope of work may alter.

Work breakdown: The project is broken down into small cycles (known as Sprints in Scrum).

Teamwork is important: Team members collaborate closely and have a clear understanding of their roles.

Iterative improvements: The work done during a cycle is frequently reassessed to improve the final output.

Collaboration with a client: A client is actively involved in the development process and can amend requirements or accept the team's recommendations.

The Agile Manifesto's four values are as follows:

• Individuals and Interactions vs. Tools and Processes.

• Working software trumps thorough documentation.

• Customer collaboration trumps contract negotiations.

• Adapting to Change trumps sticking to a plan.

12 principles that are mentioned in the agile manifesto are as follows:

1. Customer satisfaction through timely and consistent supply of useful software.

2. be open to altering needs, especially late in the development process.

3. Consistently provide working software (weeks rather than months).

4. Businesspeople and developers work together daily.

5. Projects are created on the backs of motivated people who can be trusted.

6. The best method of communication is a face-to-face interaction (co-location).

7. The ability to maintain a continuous pace of development is the most important indicator of progress.

8. Consistent development, i.e., development that can keep up with the pace of the rest of the world.

9. Maintaining a constant focus on technical excellence and good design.

10. Simplicity is crucial—the art of minimizing the amount of effort that is not done.

11. Self-organizing teams produce the best architectures, needs, and designs.

12. The team considers ways to improve its effectiveness on a regular basis and adjusts as needed.

Agile methodology benefits:

The following are some of the benefits of agile methodology:

• The software delivery is persistent in Agile.

• Working feature(s) are given to the customer at the end of each Sprint. They will be more satisfied because of this.

• Customers can examine the produced features to see if they meet their requirements.

• If customers have any input or request adjustments to the features, these can be addressed in the current or possibly next product version.

• Even in the last stages of product development, changes might be made.

• In Agile, product development and businesspeople connect daily.

• The product's design receives a lot of consideration.

Agile frameworks

Agile refers to a wide range of frameworks and methodologies that share the above-mentioned ideas and ideals. Each one has its own set of applications and distinguishing characteristics. Scrum, Kanban, Hybrid, Lean, Bimodal, and XP are the most prominent frameworks.

The three agile frameworks will be discussed in this topic: Scrum is the most popular agile framework. It is utilized entirely by 58 percent of businesses, while another 18 percent use it in conjunction with other methods. The New Product Development Game was first described in 1986 and formulated over a decade later [4]. The presentation was based on what they learned while using the strategy over the preceding few years. While Scrum predates the Agile Manifesto, it is based on agile concepts and adheres to the values set forth in that declaration [5].

Scrum aims to maintain good collaboration between individuals working on complicated projects, and it is constantly changing and adding details. It is founded on the methodical interactions of three key roles: Scrum Master, Product Owner, and Team.

A project's scrum: Master is a pivotal player. His primary role is to remove any hurdles that may obstruct the team's ability to function effectively.

The product owner: who is usually a customer or other stakeholder, is actively involved throughout the project, communicating the product's overall vision and providing timely feedback on the job done after each Sprint.

The scrum team: is a cross-functional and self-organizing group of people who are in charge of product implementation. To remain flexible and productive, it should include up to seven team members **(Figure 3)**.

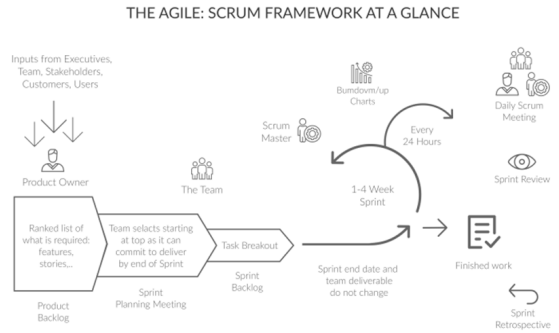

Figure 3 The agile-SCRUM framework at a glance.

Sprints and artifacts

A sprint, or fundamental unit of work in scrum, is a short development cycle required to produce shippable product iteration. A sprint lasts anything from one to four weeks: Longer iterations lack the consistency and flexibility that are key to Scrum's success. Even though there is no standard time (if it is less than four weeks), all sprints within a project should be the same length. This makes planning and tracking progress much easy. The Product Backlog, the Sprint Backlog, and the Sprint Burn down Chart are the three key artifacts used by Scrum to manage requirements and track progress. A variety of recurrent meetings, such as the Daily Scrum (Standup), Sprint Planning, Review, and Retrospective sessions, help to codify the process.

The product backlog: is a prioritized list of feature requests for the project's final product. It is a single point of contact for all requirements. As new needs, fixes, features, and details are amended or added, the product Backlog is updated.

The sprint backlog: is a list of tasks that must be completed by the team to deliver a working increment of software at the end of each Sprint. To put it another way, team members agree on which product items to deliver and how they will do it.

The sprint burn down: Chart depicts the amount of work that remains in a Sprint. It benefits both the team and the Scrum Master because it indicates daily progress and may predict whether the Sprint target will be met on time.

Scrum meetings

A variety of recurrent meetings, such as the Daily Scrum (Standup), Sprint Planning, Review, and Retrospective sessions, help to codify the process (the Sprint Retrospective) [6].

The daily scrum: is a timed meeting in which a Development Team coordinates its work and establishes a plan for the following 24 hours. The event should last 15 minutes and be held every day at the same time and place.

At Sprint Planning, the work that needs to be done is planned. This event is attended by everyone involved in the Sprint (a Product Owner, a Scrum Master, and a Development Team). They provide answers to two important questions: what work can be done and how it will be done. For a one-month Sprint, Sprint Planning takes no more than eight hours. The meeting usually takes less time for shorter Sprints. At Sprint Planning; the work that needs to be done is planned. This event is attended by everyone involved in the Sprint (a Product Owner, a Scrum Master, and a Development Team). They provide answers to two important questions: what work can be done and how it will be done. For a one-month Sprint, Sprint Planning takes no more than eight hours. The meeting usually takes less time for shorter Sprints. The team and the product owner meet at the Sprint Review at the end of each Sprint. The team displays completed work and answers questions regarding the product increment during this casual discussion. All the participants work together to figure out what they should do next to improve the product's value. For one-month Sprints, the Sprint Review is a four-hour time-boxed meeting. Retrospective Meetings are held for the entire team to reflect on their work during the Sprint. Participants examine what went well and what went wrong, identify ways to improve, and devise a strategy for putting these good changes into action. After the Review and before the following Sprint Planning, the Sprint Retrospective is held. For one-month Sprints, the event lasts three hours.

When to use scrum

Scrum is a good fit for long-term, complicated projects that require stakeholder feedback, which can have a significant impact on project requirements. Scrum may be the best option when the exact amount of work cannot be determined, and the release date is not set. Scrum has earned the trust of 89 percent of Agile users by prioritizing customer needs and on-time/on-budget delivery. As a result, the number of businesses that have adopted this strategy is astonishing. Microsoft, IBM, Yahoo, and Google are among the companies that have contributed to a public spreadsheet. According to the Scrum Alliance's newest research, Scrum has applications outside of IT. Companies in the banking, consulting, and entertainment industries use this technique to manage their work processes and improve customer collaboration. Most respondents to the 2016 State of Scrum Report (98 percent) said they would use this framework to move forward.

Kanban: comprehensive solution to handling work

Kanban is another popular project management framework. Kanbanas is one of the project management frameworks used by 43% of firms. Kanban is a simple, yet powerful, technique to producing software products that evolved from a visual system of cards used in Toyota manufacturing as a production control method. Kanban, which is Japanese for "visible signal," focuses on the visualization of workflow and prioritizes work in progress (WIP), is restricting its scope to properly match it to the team's capacity [5]. When a task is finished, the team can move on to the next item in the pipeline. As a result, the development process allows for more planning flexibility, quicker turnaround, defined objectives, and openness **(Figure 4)**.

Figure 4 The agile project management best practices and methodologies.

Kanban board: In contrast to Scrum, Kanban does not require established procedures inside the process or fixed iterations. The workflow is visualized using a Kanban board, which is often represented by sticky notes and whiteboards, or online platforms such as Trello. Trello makes Kanban more automated and digital. Because each Kanban board contains concise information about a work item, everyone on the team understands who is accountable for the item, what each person's responsibility is, when it is scheduled to be completed, and so on. To give more information, team members can post comments, send screenshots, documents, or links. Kanban teams collaborate in a cooperative manner. The ability to track progress allows coworkers to better comprehend each other's contributions to the common goal, resulting in a greater concentration on finishing the assignment well and on time.

Lean: The notion of "just in time production" underpins this software development strategy. The goal of Lean software development is to speed up the process while lowering costs.

Lean development: Remove unneeded items (anything that does not add value to the customer's project is removed), development of high quality (producing high quality in development necessitates discipline and management of the quantity of residuals produced), create Knowledge (the team is encouraged to document the entire infrastructure in order to preserve its value in the future), differentiated commitments (this point encourages the team to focus less on planning and anticipating ideas without first having a thorough understanding of the business's requirements), fast delivery (as soon as feasible, deliver value to the consumer), respecting the team (two key criteria are communicating and managing disagreements), optimize the entire process (the development sequence must be perfected enough to be able to delete errors in the code, to create a flow of true value).

Conclusion

Because software engineering is such a fast-paced industry, it necessitates flexibility and reactivity in all aspects of project development. Agile frameworks enable the delivery of cutting-edge products and the cultivation of creative experiences while maintaining product alignment with market trends and customer requirements. The efficiency of agile software development teams is critical to the success of any software project. An agile team is made up of numerous components, all of which are linked to human input when it comes to software development. These elements of agile teams are garnering greater attention in the context of productivity, and hence warrant additional investigation.

References

1. Fatema I, Sakib K (2018) Using Qualitative System Dynamics in the Development of an Agile Teamwork Productivity Model. Adv Eng Softw 11:170-85.
2. Melo C, Kon F, Conradi R, Cruzes D (2014) Productivity of agile teams: an empirical evaluation of factors and monitoring processes. SBC 4: 25-30.
3. Ambler S (2012) Roles on Agile teams: From small to large teams. Ambysoft 22: 11-12.
4. Asproni G (2004) Motivation, teamwork, and agile development. Agile Times 4:8-15.
5. Moe NB, Dingsøyr T, Dybå T (2010) A teamwork model for understanding an agile team: A case study of a Scrum project. Information and Software Technology 52: 480-91.
6. McCormick M (2012) Waterfall vs. Agile methodology. MPCS 3:1-5.