The Software Development Lifecycle  
Agile Retrospective

Katie Stapleton

Southern New Hampshire University

April 2020

Abstract

In the technological industry, the concept of the software development lifecycle is vital to understand. The software development lifecycle is the methodology in applying project management to software development. This paper focuses on one particular methodology that is gaining popularity, known the agile project management. The roles in agile, known as the product owner, scrum master, developer, and tester, are explored. After the team roles, the agile methodology and cycle is explained. Next, this paper briefly covers the more traditional form of project management, known as waterfall, in comparison to agile project management.

Keywords: Software development lifecycle (SDLC), agile methodology, waterfall methodology, scrum master, product owner, tester, developer, team collaboration

**The Software Development Lifecycle**

**Agile Retrospective**

The software development lifecycle is an important knowledge to obtain when pursuing a computer science career. The agile approach is growing in popularity. An agile team consists of multiple roles, some with unique responsibilities. The Agile methodology focuses on team collaboration, user-orientation, increasing product quality, and more. Another commonly used approach is the waterfall approach. Naturally, the agile and waterfall approach vary from one another, but a team can certainly adapt them into a hybrid approach. In this course, I took on the various roles of an agile team including applying the agile principles and tools to the project.

# Agile Team Roles

In the agile software development, a team consists of multiple roles, all of equal importance. On a standard agile team, the roles consist of one scrum master, one product owner, developer(s) and tester(s). Each role had its own responsibilities. Every role has agile principles to adhere to, with some more specific to certain roles than other roles. Tools or methods are available to uphold the agile methodology and team collaboration throughout the entire project.

The scrum master is the agile coach and peer leader of the team. The responsibilities are ensuring the team follows scrum practices, rules, and methodology while treating all team members as equals (Cobb, 2015). A scrum master provides guidance and regulations for the team to follow. Team collaboration is a vital principle for a scrum master to implement at all times. To achieve a project with the best outcome, the team needs to work together and clearly communicate with each other. Even if a product fails, the team will be able to learn from their mistakes and improve as they develop other products. A scrum master uses the practice of empirical process, which is operating of process with the expectations of frequent adaptations and inspections (Cobb, 2015). Agile functions on the principle of high levels of uncertainty and needs a process that accounts for changes, adaptations, and trial-and-error. To advocate the empirical process, a scrum master breaks the project into sprints, which are short, fixed-length time-period to complete a set amount of work to be ready for review (Rehkophf, 2020). To promote collaboration, a scrum master conducts scrum ceremonies. Scrum ceremonies are team meetings promoting face-to-face contact, teamwork, and product review that takes place throughout each spring (Cobb, 2015). The scrum ceremonies are the sprint planning, daily scrum meetings, the sprint review, and the sprint retrospective (Malsam, 2019). Scrum Master also may create project charters.

I practiced the role of a scrum master at the start of the project. I developed the team regulations regarding conduct and communication. I created the daily agenda for the daily scrum meetings. The daily agenda provides the team with simple directives regarding the discussion of the daily agenda. Every member is expected to participate and follow the daily agenda. Once given a template, I created the agile project charter for the team. The project charter states the business vision, mission statement, team roles, success criteria, project risk, rules of behavior, and communication guidelines. The charter promotes collaboration by declaring a clear set of standards to the team, with an emphasis on teamwork and a positive environment.

The product owner is mix between the role of a business representative and business analyst. The owner directly handles client relations and communicates the information back to the team. The responsibilities are maximizing the value of the product, making product decisions, providing direction on what the team needs to build, and setting work priorities (Cobb, 2015). A product owner must adhere to important principles in agile. The principles are that the product is user-oriented and the users are involved throughout the entire project. Throughout the project, a product owner meets with the client, known as the user, and sometimes actual users to discussion their expectations and obtain feedback. User stories are developed to define the requirements of the product. Each story focuses on a specific user statement in a clear and concise manner. User stories consist of the user type, priority, size (level of effort), the user’s expectations, and the acceptance criteria to meet the expectation. Though any team member can write user stories, the product owner is usually responsible for their creation. The product owner also forms and maintains the product backlog. The product backlog is a dynamic catalog of user stories, often viewed as a queue of work to be completed (Cobb, 2015). The product backlog is periodically updated, known as grooming, to keep the user stories updated, correct errors, and remove obsolete stories.

I was the product owner near the start of project, shortly after the scrum master. At the start of the project, I “attended” a client/user meeting to discuss the user’s expectations and product desires. After the meeting, I developed user stories to develop and organized the stories into the product backlog. About halfway through the product, the users wanted a change in direction from general vacations to wellness vacations. I reformed the user stories and product backlog. I would have most likely updated the team at a scrum ceremony and through email. By continuously communicating with the user and keeping the team updated, the team is able to adhere to developing a user-oriented product. Maintaining the backlog prevents confusion among the team regarding to user expectation and keeps the product development at a steady pace.

The developer creates the functionality of the product. In software development, the developer utilizes the user stories to produce features that the user is expecting of the product. The developer is to make the functionality as expected without over- or under-supplying. The developer addresses product issues and adjusts the coding as needed. A developer uses the principles of code refactoring. Code refactoring consists of removing redundancy, eliminating unnecessary functionality, revitalizing obsolete designs, and improving the current software design (Cobb, 2015). Another principles used is continuous integration. Continuous integration is the practice of continuous integrating new or altered software into the existing software and performing tests throughout the project (Cobb, 2015). A frequently used tool is user stories, usually created by the product owner. User stories allow the developer to design the code according to the user’s needs and expectation. Each story consists of the user statement, acceptance criteria and the workload of the story. Developers may need to communicate with the product owner when they need clarity and/or additional information regarding the user stories. Developers would most likely contact the owner face-to-face or through email or a messenger system. Developers also communicate with the scrum master during ceremonies to state their progress, goals, and issues they are facing.

I practiced the role of the developer near the end of the project. The project changed direction from general vacations to wellness vacations. The code was already developed for general vacations. I had to update the coding and pictures according the user’s new requests for wellness vacations. I sent an email to the product owner and tester to ask for clarity and additional information about user stories/expectations. By communicating through email, I could clearly state my issues and receive a written response. The changes could have caused confusion. Through proper communication, I collaborated with my team and maintained continuity.

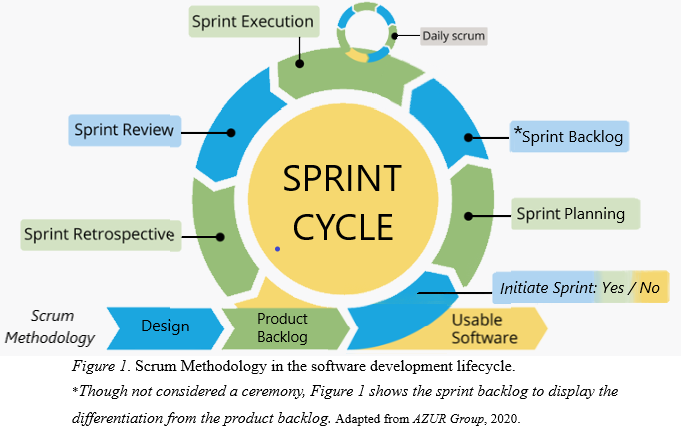
The tester is either part of the team or works directly with the team. The tester ensures the quality and reliability of the product. Throughout the entire agile project, the tester analyzes the software from the user’s perspective using test cases. One agile testing principles is value-driven testing. Since it is not possible to test the entire project, testing is performed on functions that users consider important. This allows for additional testing to be minimalized or eliminated. Another principle is automated regression testing. Testing is repeated on previous functions to ensure that that new functionalities or corrections did not negatively affect the previous functions. An additional principle is acceptance test-driven development (ATDD). ATDD is a process where all team members work out requirements, predict potential pitfalls, and reduce the change of errors before coding and throughout the project (Rouse, 2019). In agile, testers formulate pass or fall testing for the production, constructed from user stories. The pass/fall criteria is specific to the user story and consists of individualized user inputs and expected results. To pass the test, all the inputs must generate the expected results. At times, user stories may be lacking in information needed by the tester. Tester often contacts the product owner to obtain clarification on user stories, often through emails, messages, or in-person.

I practiced the role of the tester in the middle of the project. Based on the email to the product owner from the tester and the slide show jar file, I developed test cases for each user story I selected in the previous module. The test cases including titles, priority levels, dates, ID, test inputs and expected results. Once the “product owner” provided additional user feedback, I revised the test cases according to the changes in user stories and user feedback. The process encouraged collaboration because the test cases are applied to software created by the developers. The software functions and test cases are constructed based on user studies. The product owner and tester need to work together to produce the best possible outcome for the user. The tester would also work with the developer, by providing feedback on how the test cases passed or failed. The developer would then adjust the functionality as needed.

**Agile Methodology**

Agile project management is based on concept of embracing and preparing for uncertainty. Agile uses the method of rolling-wave planning. In rolling-wave planning, planning decisions are deferred to the last instant that a decision can be made without affecting the outcome of the overall project (Cobb, 2015). The process starts with developing a high level plan that defines the vision, project scope, objectives to the level of detail needed at that point of the project (Cobb, 2015). Additional plan details and requirements are developed as the project progresses. Rolling-wave planning results in better decisions because more information is available and speculation is reduced by delaying the decisions until needed. In direct relation to rolling-wave planning, agile heavily employs the technique of progressive elaboration. Progressive elaboration expects a team to learn about the product as they create it and then apply the information to produce a better product (Render, 2019). A team should not attempt to know all details about a product too far in advanced because the product will change as it develops and all the pre-planning is a waste (Render, 2019). In short, agile project management is technique utilizing high levels of uncertainty to its advantage. A high level of uncertainty is present at the start of a project, but slowly minimizes as information, development, and user communications progresses until the product is complete and satisfactory to the user(s). A common practice associated with the previous two methods is promoting simplicity and limiting the solution to “just barely good enough” to solve the issues (Cobb, 2015). A team starts with most basic and simple solution, incrementally adding features only valuable to the user. The purpose is not to shorthand the user. It is to build exactly what the user wants and prevent adding unwanted or unnecessary features.

The software development lifecycle (SDLC) is a loosely used term that can refer to multiple concepts in agile project manager. In this case, SDLC refers the framework used by the team to organize the workflow of the development and release, often known as scrum methodology (Cobb, 2015). The SDL is executed in cycles known as sprints (*Figure 1 below*). Sprints are short, fixed-length work cycles that employ agile planning and scrum ceremonies to ensure proper execution (Malsam, 2019). Before the start of the initial sprint, the project’s scope and overall design is formulated (Malsam, 2019). Next, the product owner collects user stories from the client and/or users to establish the product backlog, as previously discussed. The team then initiates and completes the sprint. When the sprint ends, the team delivers the functions created during the sprint to the user. Product errors, changes in the user’s needs, failing user satisfactions, and/or a lack of information can occur at any time. Hence, the product backlog and user stories are adjusted accordingly throughout the entire project. Sprints will keep occurring in agile projects until the product backlog is completed and the user is satisfied.

A sprint has a unique set of phases that occur during any sprint (*Figure 1 below*). At the beginning, the sprint planning ceremony takes place with the entire team. The product owner and other team members negotiate what user stories the team handles in the sprint (Cobb, 2015). The team defines the needed tasks to implement the stories and distributes the stories and tasks among the team (Cobb, 2015). The team forms the sprint backlog, which is the backlog consisting of the selected stories and tasks chosen for the sprint. Although the sprint backlog is fully established, the team updates it accordingly throughout the sprint (Rehkopf, 2020). The next phase is the sprint execution. The team developers each work on a user story(s) pulled from the sprint backlog. The developers create the necessary functions to fulfill each user story. After each user story is completed, developers often turn in software to the tester and the product owner to maintain to continuity. To help the team stay on track, the scrum masters hosts a “daily scrum” ceremony, usually at the start of each workday. The daily scrum is a brief daily meeting for the team to coordinate tasks, monitor progress, and identify any obstacles that may to impending development (Cobb, 2015). At end of a sprint, the team attends the sprint review ceremony. The team presents the finished work to the product owner for final review. Additional, the team can demonstrate their work to the user(s), stakeholders, and fellow team members (Rehkopf, 2020). The last part of the sprint is the sprint retrospective ceremony. The retrospective is a meeting for the team to review the software, discussion lessons learned, identify opportunities for improvement, and reflect upon the benefits and obstacles during the sprint (Cobb, 2015). The sprint cycle is now complete and the software developed during the sprint is furnished to the user.

In this class project, the team used the agile methodology of rolling-wave planning and progressive elaboration. The project started with a general overall and developed the specifics as it progressed. I did not have to plan the project out in full detail but rather as it progressed and I learned more. As a product owner, I attended a client/user meeting before the start of the sprint. I gathered information to develop user stories based on the user’s needs. I created a sample product backlog by developing the user stories. The product backlog provided a workload of user stories for the team to complete the product. During the sprint planning, the team was able to select which user stories to work on during the sprint. Each user story allows the team member to focus on specific functions with clear expectations and requirements. The team began the sprint execution phase, working on the user stories. As a scrum master, I created the daily agenda for the daily scrum ceremonies. It consisted of simple questions for each team member to answer. It allowed the team to keep everyone updated on their progress, prioritize their work, and address any obstacles on a daily basis. As a tester, I created test cases to test if the product met the requirements of the user stories. The test cases made it clear what passed and failed, allowing other team members to make corrections and successful fulfill the story. Continuous communication during the project ensured team collaboration. As a developer and a tester, I sent out emails to the product owner when I needed more information regarding the user stories. Collaboration and communication was especially important when the project was interrupted in the middle of its design. The user decided to switch its needs from developing a website promoting their general vacation to a website specifically promoting wellness vacations to attract customers. At that point, some product functions were designed for general vacations, not wellness vacation. With poor communication, the team could have accidently developed an “expired” product for general vacation. With poor collaboration, the product could have ended up skewed trying to meet to different missions. However, agile promotes planning for changes and expected the unexpected. After each sprint, the team reviews the products and updates the stories based on the user’s statements and feedback. Product functions are individually developed during a sprint and can be adjusted, even in later sprints. By maintaining communications through the entire project and every sprint, the team was able to stay updated with the user and each other, adapt the user stories accordingly, and product the produce the user needed.

**Agile vs Waterfall Approach**

More than just the agile approach exists in project management/software developer. Although the agile approach is gaining popularity, the waterfall approach is different approach often used by companies. (Cobb, 2015). Roles and principles can vary, sometimes to high degrees, between the agile and waterfall development approach. However, a team does not have to be 100 percentage agile or waterfall. A team may need to consider which methods from each approach best suites their environment, leading to a hybrid approach of using methods from both agile and waterfall.

Waterfall project management is plan-driven from start to finish. The strategy attempts to reduce the project to a low-level of uncertainty before the project starts. It provides established production time lines, predictable estimates, and product features before the start of the project. A product manager(s) plans the entire project in detail including product requirements, every phases in the project, and distinct goal for each phase (Cobb, 2015). When the manager cannot find a clear answer, assumptions are often made to counter uncertainty. Development is sequential with no turning back after a phase is completed (Cobb, 2015). Product testing and repair does not occur until the end of the project. Agile project management uses a much more adaptive approach. The strategy is to plan for high levels of product uncertainty, promoting real-time decision-making, and having small teams working together intensively (Cobb, 2015). Agile focuses on providing value and functionality to the user(s) and obtaining user input/feedback as the project advances (Cobb, 2015). However, it can be challenging in agile to produce pre-production time and cost estimates, and a complete list of product functions and descriptions. Instead of having a project manager, the entire team handles the project each taking on various duties. Remember that despite the differences between the two approaches, a company can develop their own hybrid approach based on their needs and other factors. Overall, the key factors to consider are the level of uncertainty, the capabilities of the project team, and information demanded by the user before the project starts (Cobb, 2015).

In this college course, I participated in a project for a fictional travel company, known as SNHU Travel. For this project, the selected approach was agile project management. The project consisted of a higher level of uncertainty, low user demands at the start of the project, and the full capabilities of the team to complete their duties. For example, the company desired to expand their website to attract new customers and increase sales. However, the user had varying ideas on what they desired in the final product. Next, SNHU Travel did not require a detailed budget statement, a solid features list, or hard deadlines for each phase, and seemed open to communication throughout the project. Third, the team seemed like a well-trained scrum/agile team with a decent amount of experience. Each team member had his or her own role. The team showed professional communications when addressing issues or changes along with keeping everyone updated.

The employee roles are different in the agile and waterfall approach. The waterfall approach uses the role of a traditional project manager to handle the project. A traditional project manager provides clear, directions instructions for all team member on how to handle the project and each phase. However, the project manager does not define the requirements of the product and does not perform product testing (Cobb, 2015). Notably, product testing and quality review does not occur until the end of the project. The agile approach focuses on teamwork, with the roles of scrum master, product owner, developer(s) and tester(s). A scrum master is a peer-coach that facilitates the team, providing minimal direction and is a non-existent role in waterfall. The product owner is hybrid of a project manager and business analyst. The product owner defines the requirements, serves as the primary communicator with the client, and provides the business aspect in the project. An agile tester is involved throughout the entire project instead of the very end. The agile tester tests the product and provides feedback on user-valued functions. An agile developer has a greater flexibility when designing a product. Errors are addressed as the project progresses. As a result, the developer can test their own ideas that may fail initially, but still resolve the issue quickly. The agile approach does not use traditional project management, but applies team members to handle various project management duties.

**Personal Reflection**

With almost no professional career experience, it is hard to understand and even harder to decide on the numerous roles available in the market. Before I started this class, the professional role I planned to pursue was simply a software engineer or software developer. I would mostly start as a software developer and then transitions in a software engineering position. For my long-term career, I hope to obtain a senior-level or management position in the computer science or software engineering field.

I have not held any information technology, computer science, or software-related positions in my career history. Recently, I was exploring career websites, such as Linked In, to see what roles people were pursuing and what types of jobs were currently available. Certain individuals had the titles of scrum master and product owners, which I found confusing. At that time, I did not have any knowledge of the agile methodology and its relation to software development. Now, I finally understand the roles of product owner, scrum master, and an agile software developer/team member. Although I am mainly interested in starting in software engineering or development, the position of a scrum master seems like an appealing career avenue also. I would consider the position after some experience and income as a developer or engineer. I believe I have the skills the skills to become a successful scrum master if I would choose to.

Before I become a scrum master, I would like to gain some experience or better familiarity in agile estimation. I want to be able to read and write user stories with a good understanding of the level of effort involved in the story. I would be able to have a higher comprehension of the risks, uncertainty, and level of effort that each team member is tackling with each user story. Understanding what is taking place within your team is important for assessing productivity and promoting constructive team dynamics. I would also like to gain a more in-depth knowledge of sprint planning and spring retrospectives. I want a better understanding of what I need to examine, calculate, evaluated, and report.

I believe that the agile project charter exemplifies me in the role of a scrum master. A charter is tool that provides a clear written statement of the team’s roles, project goals, and professional expectations. The charter provides team member a strong assertion of properly communicating with each other. It is not referring to how to send messages, but instead how team members are to conduct themselves when communicating with others, which should be in a positive and constructive manner. Specifically, the example provided in class consisted of the business vision, mission statement, team members and their roles, success criteria, key project risks, rules of behavior, and communication guidelines. I pursed a business degree and worked in business administration before starting my computer science degree. I am fully aware of the importance between treating team members as equals, providing constructive communication, and embracing unexpected challenges. When the need arises, I am not afraid to step up and become a leader.

To help achieve my professional goals, I need to continue pursuing my computer science degree. If possible, I may continue my education if I can find an employer who pays for or reimburses education related the company or position. The classes may be graduate degree or short training sessions, most likely related to understanding scrum, software development, communications, and agile team work. For any technology career, an individual should consider obtain official licenses pertaining to their field and/or the field they wish to pursue. The cost of a becoming a certified scrum master averages about at $1000 to obtain the license. I realize the license is not a necessarily a requirement, but higher-paying companies usually expected licenses in any technological role. If I become a software engineer or developer, I would need to examine licenses related to programming and coding. I would also consider learning more about the waterfall approach. Not all companies use the agile approach. I may take a college course, go to a seminar, or simply read book to learn more.

References

AZUR Group. (2020). Agile Solution - Optimized Agile Solution in 3 Phases. Retrieved April 18, 2020, from https://www.cloudapplicationdev.com/en/our-optimized-agile-solution-in-3-phases/

Cobb, C. G. (2015). *The project managers guide to mastering agile: principles and practices for an adaptive approach*. Retrieved from <https://mbsdirect.vitalsource.com/#/books/9781118991770/cfi/0>

Malsam, W. (2019, July 27). A Beginner's Guide to Scrum Ceremonies. Retrieved from https://www.projectmanager.com/blog/guide-to-scrum-ceremonies

Rehkopf, M. (2020). Agile coach: sprints. Retrieved April 17, 2020, from https://www.atlassian.com/agile/scrum/sprints

Render, J. (2019, June 17). What is Progressive Elaboration. Retrieved from https://agile-mercurial.com/2019/06/10/what-is-progressive-elaboration/

Rouse, M. (2019, July 24). What is Acceptance Test–driven Development. Retrieved from <https://whatis.techtarget.com/definition/acceptance-testdriven-development-ATDD>