How to Choose your Agile Framework

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Jacob Rice

Spring, 2022

Rosanne Vrugtman, Department of Computer Science

How to Choose Your Agile Framework

CS4991 Capstone Report, 2022
Jacob Rice
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
jwr5ky@virginia.edu

ABSTRACT
Choosing the appropriate agile framework for your software project or organization can be difficult. The choice depends on many factors, both organizational and technical. To simplify the process, I created a set of guidelines to keep in mind while selecting the best approach from three of the most common agile frameworks: Scrum, Kanban, and Extreme Programming (XP). While my recommendations provide a sufficient starting point for the simplification of the selection process, a more thorough and nuanced framework could be created from this foundation.

1. INTRODUCTION
When developing a software product, it is important to determine the processes by which the software will actually be developed. By doing this early in the process, it is possible to resolve several managerial concerns—knowing when a task is finished and what to do next [1]. Additionally, creating standardized processes for the development of the software product makes it easier for multiple people on a team to work together because all team members will be making decisions based on shared knowledge.

According to the 15th State of Agile Report [6], 94% of companies surveyed reported that they used Agile processes. This means an enormous proportion of development that is being done via Agile development. Thus, it is necessary to understand how this 94% is selecting processes and whether those processes are appropriate for their situation.

Further, these companies also practice a number of different kinds of Agile Frameworks. The same State of Agile report explicitly denotes the usage of five unique frameworks: Scrum, Kanban, Iterative, Xtreme Programming (XP), and Lean Startup. Further, the report mentions two frameworks which, are actually compositions of other frameworks that have evolved into separate frameworks: Scrumban and Scrum/XP hybrids. With all of these differing frameworks, it can be difficult to select the appropriate framework for your team. Thus, it becomes necessary to understand when and how each framework should be used so that the most appropriate framework is selected.

2. BACKGROUND
Because there are so many different types of Agile frameworks to choose from, it is important to understand exactly how to select the most appropriate framework for a given use. In order to do so, one must understand what organizational factors must be investigated, as well as the importance of each of them. If not adequately investigated, the implementation of Agile may fail due to conflicts in the organization culture either, with the methods selected or with Agile as a whole [2]. Further, selecting the wrong framework can reduce the quality of work produced, and the team's morale [3].

3. LITERATURE REVIEW
The organizational factors one must consider when selecting a framework are many and varied. For instance, according to their Rand's 2021 article, one must consider the objectives of the project [4]. In addition, Crouch says that when selecting an Agile framework, you must consider the nature of the work and the culture of your organization [5]. Thus, the types of considerations one should make when selecting a framework tend to fall into one of two categories: the nature of the work and how your developers work. In addition to these considerations, one must also consider whether one is switching from a waterfall style to Agile.

The first consideration is the overall style of work of your developers. When selecting a framework, consider how the developers work, then select a framework that highlights the developer's strengths and shores up their weaknesses [4]. For example, if your developers are highly collaborative, select a framework that allows them to embrace that collaborative workstyle. On the other hand, do not enforce excessive collaboration if the team prefers to work relatively independently. In this way, it is possible to avoid the risk reducing morale and work quality. If switching from a more waterfall-based development system, avoid frameworks that fall back into the waterfall style, like Rational Unified Process (RUP) or Scaled Agile Framework (SAFe) [5].

Additionally, one must consider the nature of the work being done.~~,~~ For projects in which dependency management is complex, more structured frameworks with increased planning may be beneficial. [5]. On the other hand if your project is more error-tolerant priorities are more fluid, it may be worth considering a framework which allows the project to move at a faster pace, with less structure and planning.

4. METHODOLOGIES

I investigated the three most common agile frameworks that are not compositions of multiple frameworks according to the 2021 State of Agile report, in order of total usage percent: Scrum, Kanban, and Extreme Programming (XP) [6]. By looking at these five frameworks, I created a set of guidelines for selecting the best approach from this subset of frameworks.

4.1 Scrum

Scrum is the Agile framework most employed across industry [6]. Scrum is a "lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems." [7]. Scrum is "intentionally vague," though it is defined by several core values: commitment, focus, openness, respect, and courage. Even though the framework is made to be flexible, it nonetheless defines four key events that are important to the overall scrum process.

The first event is the sprint. The sprint is an event which takes place over a fixed length period of time and repeats once the previous sprint ends, in which work is done toward completing a "sprint goal." The second event, known as daily scrum, is an opportunity to discuss overall progress towards the sprint goal and adjust backlog. The third, the sprint review, exists to determine how successful the sprint was and to adapt future sprints if need be. In the final event, the sprint retrospective, developers come together to discuss how the sprint went and to identify what can be improved in future iterations of the process.

In general, the iterative nature of Scrum promotes a lot of flexibility in how work is handled. This flexibility also leaves room to implement additional processes as needed. Through its values and the events it implements, Scrum promotes communication and collaboration between small groups of developers. In addition, this framework sells itself as having a "flat" hierarchy, in which responsibilities are evenly distributed. The scrum framework would work well with teams of developers who value communication, flexibility, and equally distributed power in their work processes.

The sprint-based nature of Scrum means that planning is necessarily short term. This provides Scrum with much of its flexibility. However, it also leaves the potential for code that is poorly planned in the context of the larger system and full of technical debt. Of course, these issues can be fixed in subsequent sprints, but this is still potentially an issue to be considered when choosing to work with this framework.

4.2 Kanban

Originally created by Toyota for use in its manufacturing facilities, Kanban is a framework which has since been adapted for software engineering work [8]. Kanban is founded on the basis of several key principles which can be separated into two categories: changing management principles, which entails incremental change as well as the encouragement of leadership at all levels, and service delivery principles, which emphasizes the regular review of services to ensure they meet customer needs.

The Kanban system commonly employs several practices. The first is to visualize the workflow, usually using a board. The second practice is limiting of work in progress, which allows developers to focus their efforts more and identify bottlenecks. This also ensures that work flows smoothly from stage to stage in the development process. Additionally, those who implement Kanban often implement feedback loops to ensure that work is being done efficiently and to make changes to the process if needed.

Kanban has the potential to succeed in organizations where change is hard to make, because it is initially rolled out incrementally. Any changes to the process after implementation are also done incrementally to ensure that they work. Further, the limiting of in-progress work helps to focus the efforts of developers—useful in environments in which developers have historically been disjointed in the work being done, and preventing the team from diverging too far from goals. Because of this focus, Kanban may be useful for projects in which the number of concurrent tasks needs to be limited. For example, sysadmins use Kanban to maintain focus on primary tasks while performing miscellaneous tasks as they arise [9]. Kanban offers a powerful way for developers to manage and visualize workflows while focusing attention on more manageable subsets of work.

4.3 Xtreme Programming (XP)

The XP framework greatly emphasizes the technical aspect of software development, much unlike Scrum and Kanban. The XP process consists of steps: planning, designing, coding, testing, and listening (customer feedback) [10]. Many of the processes in XP also emphasize the production of high-quality code via the practice of test-driven development (writing tests before code) and pair programming.

The framework also places high value on constant communication, both among developers and with the customer. In fact, one relatively common practice is to have an onsite customer to whom developers can pose questions.. This emphasis on communication can also be seen in the last phase of the process—listening. In the listening phase, the developers conduct evaluation of the current iteration of the product to verify customer approval of the product.

The XP framework emphasizes the production of quality code and constant communication with the customer to ensure that developers create high quality products. The emphasis on communication with the customer results in a system which guarantees customer satisfaction. Additionally, the practices of XP also necessitate a highly collaborative workplace. Pair programming, in particular, requires developers to work together simultaneously on the same piece of code to ensure adherence to code quality principles.

XP's emphasis on quality code lends itself well to types of work where code must be of high quality. For example, flight control systems must be bug-free, or as close as one can reasonably get to it, or one risks disasters such as the crashes of two Boeing 737 MAX planes due to faulty software [1]. Through the use of test-driven development, XP gives a high degree of certainty that code produced in the constraints of the framework will be free of major bugs.

5. Proposed Guidelines

These guidelines consist of two steps: identifying the organization's culture; and identifying the work that is being done and the values that must be embodied to successfully do this work. These guidelines, then serve as a quick reference to compare against the culture and work done for a given software product so that the best framework can be selected for a given use case.

5.1 Organizational culture

- ***Scrum:*** collaboration, transparency, communication within the team, small teams, flat hierarchy
- ***Kanban:*** Focus on efficiency, resistance to change, maintenance of hierarchy
- ***XP:*** highly collaborative, highly communicative, both between customer and developers.

5.2 Nature of Work

- ***Scrum:*** work needs to be done quickly and iteratively with less emphasis on code quality and more on pushing features
- ***Kanban:*** need to limit concurrent tasks and monitor what is being done
- ***XP:*** work needs to be done iteratively with high emphasis on code quality

If there is a mismatch between the framework implied by these two steps, one must decide whether it is more important to maintain the cultural norms of the organization or to align the processes to the nature of the work being done.

6. CONCLUSION

Selecting the appropriate Agile methodology for your software challenge can be a daunting task requiring investigation of your organization's culture and work. By creating this set of guidelines, I have provided a brief framework to assist in the selection of these frameworks. This reduces stress on those involved in the selection process by providing guidance on the kinds of projects and work cultures in which each of the methodologies is most effective.

7. FUTURE WORK

The guidelines produced in this paper are not comprehensive. Only a small number of many methodologies were examined. Some of the major frameworks like Scrumban and Lean Development are not covered by this paper. Thus, this set of guidelines could be expanded to improve these and other frameworks not considered. Additionally, practices from different methodologies are often combined in the workplace to accommodate the complexity of work and workplace culture. Thus, it is important to further expand these guidelines to cover the cases in which elements from different frameworks should be composed together.

REFERENCES

1. Mark Sherrif and Paul McBurney. n.d.. Software Process and Methodologies. (N.d.) Retrieved December 12, 2022 from https://docs.google.com/presentation/d/1JW2Ci1oqMJyYIirwo-VaKeC_4BujxM0eKQoNGlQuMs8/edit#slide=id.g3267a0cfde_0_122
2. Gloria Miller. 2013. Agile problems, challenges, & failures. In *PMI® Global Congress 2013*. (New Orleans, USA), PA: Project Management Institute.
3. Mike Cohn. 2021. What Happens When You Use Agile for the Wrong Reasons? (January 2021) Retrieved December 1 2021 from https://www.linkedin.com/pulse/what-happens-when-you-use-agile-wrong-reasons-mike-cohn/.
4. B. Rand. 2021. What Are Agile Frameworks? (Plus How To Choose One). (May 2021). Retrieved December 1, 2022 from https://www.indeed.com/career-advice/career-development/agile-frameworks#:~:text=Understanding%20the%20desired%20outcome%20of,can%20help%20guide%20your%20decision.
5. Alan Crouch. 2020. Selecting the Right Agile Framework. (March 2020). Retrieved December 1, 2022 from https://www.techwell.com/techwell-insights/2020/03/selecting-right-agile-framework
6. Digital.ai. 2021. 15th Annual State Of Agile Report. Retrieved December 1, 2022 from https://digital.ai/resource-center/analyst-reports/state-of-agile-report/
7. Ken Schwaber and Jeff Sutherland. 2020. The Definitive Guide to Scrum: The Rules of the Game. (November 2020). Retrieved December 1, 2022 from https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100
8. Kanbanize. 2022. What Is Kanban? Explained for Beginners. Retrieved December 1, 2022 from https://kanbanize.com/kanban-resources/getting-started/what-is-kanban
9. Chirs Peeters. 2020. The (Sysadmin) Kanban. (August 2020). Retrieved December 1, 2022 from https://chrispeeters.net/blog/the-sysadmin-kanban/
10. Altexsoft. 2021. (January 2021). Extreme Programming: Values, Principles, and Practices. Retrieved December 1, 2022 from: https://www.altexsoft.com/blog/business/extreme-programming-values-principles-and-practices/
11. Matt Hamblen. 2020. (March 2020). Killer software: 4 lessons from the deadly 737 MAX crashes. Retrieved December 1, 2022. https://www.fierceelectronics.com/electronics/killer-software-4-lessons-from-deadly-737-max-crashes