A STUDY ON USING WATERFALL AND AGILE METHODS IN SOFTWARE PROJECT MANAGEMENT

Bogdan-Alexandru Andrei[1]
Andrei-Cosmin Casu-Pop [2]
Sorin-Catalin Gheorghe[3]
Costin-Anton Boiangiu[4]

ABSTRACT

There are many project management methodologies which one can choose from when starting a new project. The purpose of this article is to analyze the most known methodologies, Agile and Waterfall, in order to determine which is the most suitable for a software project.

We propose a practical study by analyzing the results of a survey designed to capture the experience of developers with the aforementioned methodologies. We will focus on the Scrum and Kanban Agile methods and Waterfall to analyze the findings of the study.

Given the results of the study, we concluded that there is no silver bullet solution when it comes to choosing the methodology for a project, as numerous factors need to be accounted for. Waterfall will be a better solution for small projects that have well-defined requirements that will not change, while Agile is preferred when continuous delivery and feedback are important, requirements are not well defined and time to market is more important than releasing a full feature version.

KEYWORDS: ***Agile, Scrum, Kanban, Waterfall, Project Management***

INTRODUCTION

Kanban [1] was first introduced as a scheduling method for assembly lines in Toyota factories. It was developed to improve the workflow and to maintain a high level of production. Despite its modest roots, it was quickly acknowledged worldwide as an efficient way to "organize" projects. Kanban was transformed into an abstract concept that could be applied to different sectors and industries, thanks to its efficiency.

Kanban is composed of five key stages: *visualizing the workflow*, *limiting work-in-progress*, *managing the workflow*, *making each and every step unambiguous* and ultimately *evolving as one single and precise mechanism*.

The aim of Kanban is to remove any "bottlenecks" from a streamlined process and to maximize efficiency and collaborative teamwork across the whole team. This goal has led

[1] Student, alex.bogdan.acs@gmail.com, "Politehnica" University of Bucharest, 060042 Bucharest, Romania
[2] Student, casupopandrei@gmail.com, "Politehnica" University of Bucharest, 060042 Bucharest, Romania
[3] Student, gheorghe.catalin97@gmail.com, "Politehnica" University of Bucharest, 060042 Bucharest, Romania
[4] Professor PhD Eng., costin.boiangiu@cs.pub.ro, "Politehnica" University of Bucharest, 060042 Bucharest, Romania

to the creation of the Kanban board, which holds 4 different sets that should describe any task affiliated to the current project:

- ***Ideas*** – these are tasks that are still in a state of ambiguity and there are multiple discussions between members whether the task or feature should be implemented or not ("is it too time-consuming?", "is it profitable?", "what are the pros and the cons?")
- ***To do*** – these tasks have passed the first stage and it is clear that they should be implemented. The assignment of the task remains the only problem to solve. Kanban tries to improve the workflow by assigning tasks such that there are no "blocked" members, which could refer to either waiting for another person to finish their share of the work or to having too many "to do" tasks simultaneous
- ***In progress*** – the task has been assigned and is actively being developed by a member of the team
- ***Done*** – the task has been completed and there is no more work to be done regarding this particular feature

Kanban still imposes "a tricky part" of defining when a task is considered to be done. It is natural that with so many fields and different types of projects, tasks can get quite diverse, which could mean potential problems in knowing when a task is done. This needs to be discussed thoroughly in the "idea" stage, to set clear and unambiguous goals, which can objectively be reviewed at the end.

The Waterfall model [2][3] is the first applied software development strategy, resembling the designs that were used in other industries. This strategy allows for a project to be split into multiple fixed phases, with each phase requiring the analysis and work from the previous phase:

- **Requirements** – analyzing business needs and extensive documentation of all features
- **Design** – choosing all required technology and planning the full software infrastructure and interaction
- **Coding** – solving all problems, optimizing solutions and implementing each component described in the **requirements** phase, using the diagrams and blueprints from the **design** phase
- **Testing** – extensive testing of all implemented features and components and solving any occurring issues
- **Operations** – deployment to a production environment

The Waterfall model assumes that once the initial requirements are set and every goal has been cleared of any ambiguities, there is an unobstructed road which the development team will follow towards finishing the project. However, in most real-life cases, this is not true, as customers can change their opinion towards different features, in which case some, if not all, the phases will have to be re-evaluated. This attracts additional costs and time spent on different parts of the project, which in turn could lower customer satisfaction. This is the most obvious flaw of the Waterfall model, but it does not mean, this strategy should never be applied.

For example, when working on small projects with fixed deadlines, budget, and scope, the Waterfall model could help a team organize better, as every member would have

extensive documentation from the beginning and the experience and knowledge lost if a member leaves the project is minimized. Time can be distributed optimally since every phase has a fixed period of time allocated, but in the eventuality that one task is late, all tasks will be late.

In the end, the product will have better cohesion since the design has been completed from the first phase and everything has been taken into account. This makes for an easier streamlined process for the team, in exchange for any flexibility regarding future changes and additions.

Scrum [1][4] is a framework for developing complex projects and organizing work, based on a set of values (courage, focus, commitment, respect, openness), principles and practices which provide a foundation for all members. The Scrum framework offers great flexibility and versatility when facing changing constraints, whether financial or technologically and its key for success is always targeting the highest-priority tasks, each of these going through a process of 7 steps: ***requirement*** **elaboration, design, development, comprehensive testing, integration, documentation,** and **approval.**

The core of the Scrum framework is composed of 3 roles, 3 artifacts, and 5 events.

Scrum Roles:

- product owner - is responsible for deciding which features should be implemented and in what order. The product owner will maintain communication with all other members and is responsible for the solution that is being developed.
- development team - the members from the development team are traditional software developers which focus on multiple domains (architect, tester, developer, UI/UX). The development team will be self-organized and will decide how they should proceed in order to achieve the goals set by the product owner in the most efficient way possible.
- scrum master - is a leader that helps the rest of the members understand and correctly apply the Scrum framework. The scrum master should not be confused with a project manager, as they cannot exert any control over what the development decides, but the scrum master can help by protecting the team from outside interference.

Scrum Artifacts:

- product backlog - a prioritized list of items that need to be solved. The backlog can contain more than simple tasks or features, such as changes to existing components, “bugs” or errors that need to be fixed, infrastructure improvements and so on.
- sprint backlog - a prioritized list of items that should be solved during the respective sprint (it is essentially a product backlog, but much more lightweight, since the items should be addressed during a relatively small amount of time, opposite of the product backlog which could represent work for several months)
- increment - a collection of the product backlog items that were solved in a sprint.

Events:

- Sprint - when using a scrum framework, work is performed in iterations that are time-boxed, and are almost never longer than 30 days. The sprints have the same

time duration and are dedicated to implementing one feature without any distractions or goal-altering interventions.

- Sprint planning - represents a meeting for creating the sprint backlog and fixing the goals for the current sprint, while also clearing any ambiguities. The product owner and the development team agree on the sprint goal, and, taking said goal into account, the development team will order the sprint backlog accordingly.
- Daily scrum - represents a daily meeting, usually lasting no longer than 15 minutes, and occurring at the same time every day, which is not focused on problem-solving, but on what was accomplished since the last daily scrum and what needs to be prioritized for the respective day. It has the role to synchronize the work between team members and for management to be updated on the status of the project.
- Sprint retrospective - the essence of the sprint retrospective is to objectively look at what was accomplished, and to discuss what went accordingly and what did not, with the goal of figuring what can be improved for feature sprints. By the end of the Sprint Retrospective, the Scrum Team will identify improvements that should be implemented during the next Sprint, a process which proves the adaptability of the Scrum framework

PREVIOUS WORK

We have many theories and evaluations of different proposed models for software development [5][6], each having strengths and weaknesses. These strategies and frameworks are continuously updated spawning new hybrid models or improvements, such as the Sashimi model, developed by Peter DeGrace. The Sashimi model is a modified version of the waterfall model, with the key feature that development phases can be overlapped. The idea the model is based on is identifying errors early on during the development phase. This adds an iterative feature to the model for better flexibility and minimization of “bugs” and errors overall. The time between the detection of an error and the actual solution to that error is reduced, ultimately reducing the time spent during the testing phase and producing higher quality software.

Another important feature of the Sashimi model is treating the documentation as a unified entity, instead of exchanging documentation between different teams responsible for different phases, which leads to a better cohesion and a reduced volume of documentation. This model also has its own weaknesses such as unclear key development milestones, the difficulty of synchronizing team members and increased difficulty in monitoring individual activities.

PROPOSED APPROACH

In this research, a comparison will be made between the four most popular methodologies, while also trying to find the preferences of students and entry-level developers. As it was described in the Introduction of the document, we will compare two Agile subsets (Scrum and Kanban) and then the general modern Agile style with the classical Waterfall.

Scrum vs Kanban

Even if both methodologies are Agile types, they are different in regards to how strict a project has to be done. Kanban will allow teams and their members organize more freely. From this point of view, the following differences will be considered during the research:

- setting up roles in a team is not a requirement
- meetings are not restricted by time-boxed iterations (e.g. sprints)
- the board will be continuously updated and stories can be added anytime if they fit into the current workflow
- any member or team can be the owner of the board
- estimates for tasks are not needed(e.g. time, user points)

There are also a number of similarities that must be taken into account:

- both methodologies must be tested before a team can decide what approach to adopt
- focused on fast delivering functionalities
- both will discuss frequently with the customer in order to obtain feedback as soon as possible (being transparent with their work)

Choosing one of these methodologies will be a team decision, based on their style and on the type of project.

Agile vs Waterfall

Considering the aforementioned characteristics of Scrum and Kanban, we can identify their similarity, being just two different Agile ways to manage a project. A method considered much stricter than Agile is Waterfall. Comparing Waterfall with the previous methodologies will result in a number of clear differences. The Waterfall methodology is characterized by:

- having a structured process, each step of the project development has to be completed sequentially
- completed steps cannot be modified
- the project will begin with defined requirements that will not suffer any changes during the project development

Agile and Waterfall are completely different methodologies and their only common goal is to deliver a high-quality product.

Developers opinion

In order to study the preferences of students and junior developer, a form was created and distributed. The form answers will reflect what methodology fits for each candidate, based on their previous experiences *(as stated before, choosing a methodology should be a team decision based on their needs).*

In order to determine the experience of the candidate, they will be asked them to state how many years of experience they have. The main interest is in the first category (less than 2 years), as it will show how students and junior developers interact with the chosen methodologies.

After determining how much work experience they have, they will specify what methodologies they used and how they interacted with it. If the candidate used a

methodology, a set of questions will be given to evaluating how they worked with it. The following metrics are of interest to the research:

- ***The number of people involved in the project*** – an estimate of the number of persons involved in the project, used in studying how fit the methodology was for teams with a small or a big number of members.
- ***Duration of the project*** - the duration of the project, used to determine how each methodology fit for long and short term projects
- ***Project completion*** –whether the project they worked on has been completed or not. This metric can measure the efficiency of each methodology for the given project.
- ***Project success*** –an estimate on how successful the project was, used to determine if the chosen methodology fit for the requirements and the resources of the project.
- ***Personal satisfaction*** – how satisfied the participants were with the chosen methodology. This metric was used in order to determine which approach is preferred by students and junior developers. In the end, the candidates were asked for pros and cons of using said methodology on their project.

STUDY RESULTS

The form was sent to 32 candidates in order to examine their experience with the presented methodologies. A set of questions was chosen in order to evaluate in what circumstances the candidates used Scrum, Kanban, and Waterfall.

The research focused on students who are at the beginning of their career in software development. Most of our candidates were chosen based on their experience (< 2 years). The main reason for targeting this category is to gather relevant data from young developers attending their first internship (or internships) and finding their favorite project management methodology.

The majority of the participants had less than 2 years of working experience, consisting mainly of students and junior developers, representing 84.4% of the candidates. The results regarding the working experience of the participants can be visualized in *Figure 1*.

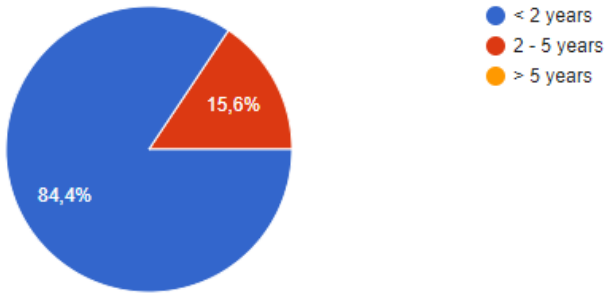

Figure 1. Years of experience

The most popular methodology used by the candidates was Scrum, with 62.5% of them using Scrum at least once. 28.1% of them used Waterfall and only 25% used Kanban. The number of people who have encountered each methodology can be seen in *Figure 2*.

By asking the candidates the size of their teams for the chosen project, it was discovered that the numbers were between 2 and 15. While most Scrum teams had 4, 5 or 7 members,

the Kanban teams usually had 5. Waterfall projects had even smaller teams, of 3 or 4 people. The number of members for each team, reported by our candidates, can be seen in *Figures 3, 4, 5*.

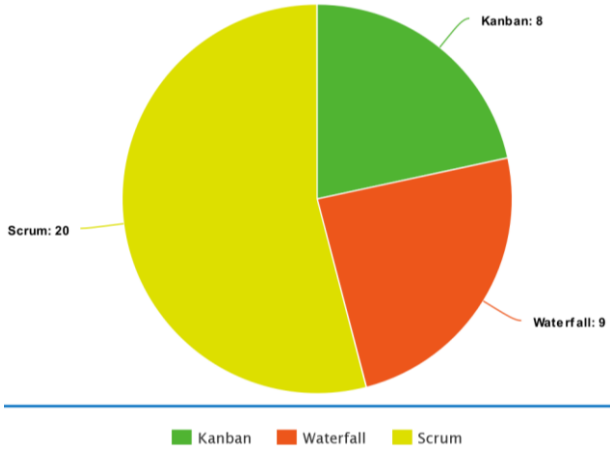

Figure 2. Number of candidates for each methodology

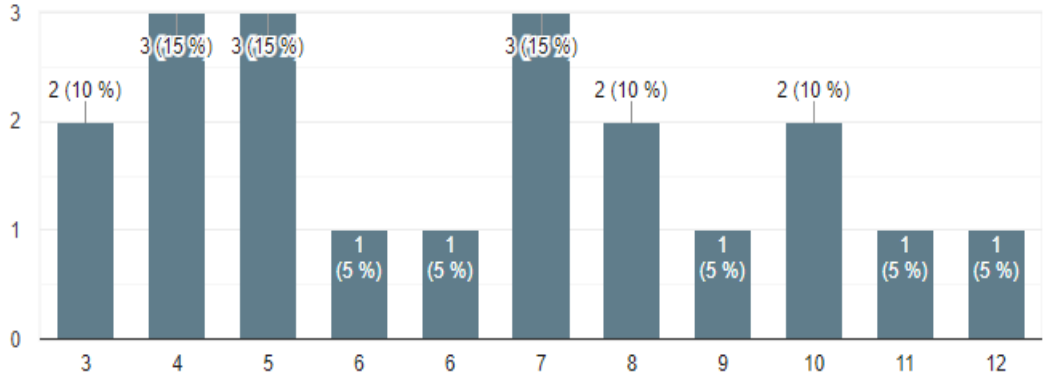

Figure 3. Number of team members for Scrum teams

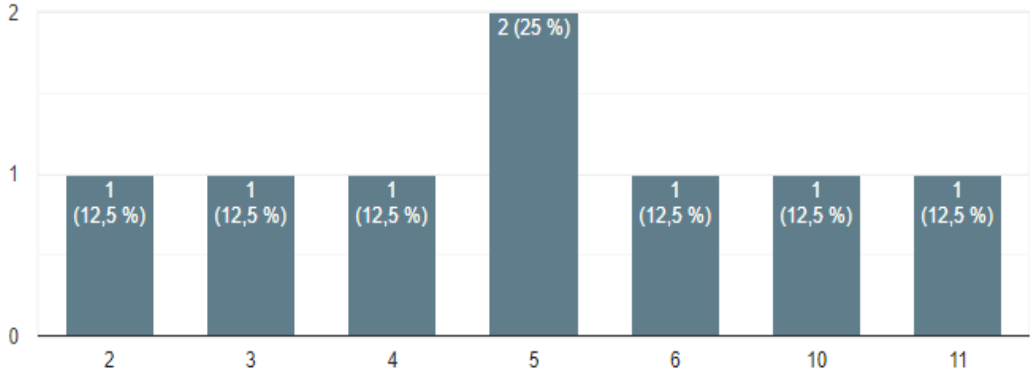

Figure 4. Number of team members for Kanban teams

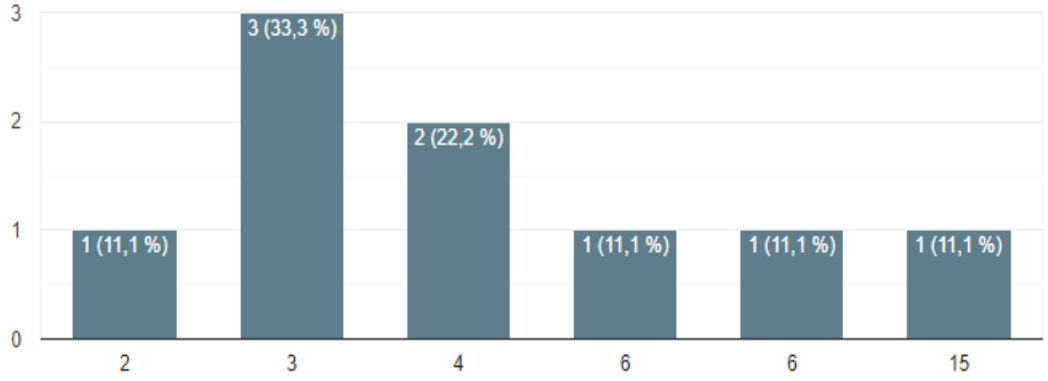

Figure 5. Number of team members for Waterfall teams.

Most of the candidates participated in short term projects, lasting less than 6 months. Waterfall presented the shortest duration, with 88.9% of candidates stating that the duration of their project was less than 6 months. Kanban was used for projects with variable length, from less than 6 months to more than 1 year (*Figure 6*).

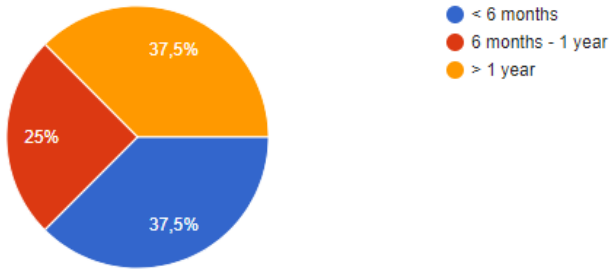

Figure 6. Kanban projects duration

The majority of the Scrum projects were short term projects as well, with 70% of the respondents affirming their project lasted less than 6 months (*Figure 7*).

All of the projects that used Waterfall were finished. Similarly, 95% of the ones that used Scrum has been finished. A high percentage of projects that used Kanban were abandoned, as seen in *Figure 8*.

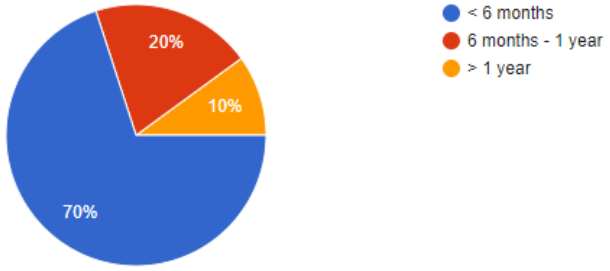

Figure 7. Scrum projects duration

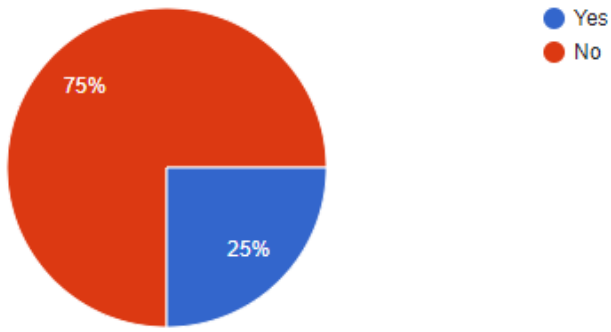

Figure 8. Finished Kanban projects

Candidates considered that Scrum delivered the most successful projects, with 50% of them classifying the success of the projects as 4 and 5 on a scale from 1 to 5. Kanban results are close to the previous ones, with 50% voting for 5 and 37% for 4. Most of the Waterfall users estimated the success of the project as a 4, with 55% of the votes. A discrepancy can be seen between the number of abandoned Kanban projects and the success of them. The possible explanation for this is that most candidates used Kanban for projects designed to be experiments.

While the big majority of Scrum and Kanban users gave a score of 4 and 5 (90% and 87.5%, respectively), Waterfall has with an average of 77% votes for 4 or 5. However, most of the votes were scores of 4. Overall, all candidates enjoyed using each methodology.

Respondents that used Scrum reported that they were highly motivated during the development and liked the fact that they were in good synchronization with the rest of the team. However, some of them did not enjoy the high number of meetings and the fact that they induced a routine.

Kanban users enjoyed the simplicity and flexibility of the methodology while reporting issues with task prioritization and the amount of time they spent updating the cards.

Waterfall users enjoyed the Plan-Driven Development that this methodology encourages and the strict requirements imposed from the beginning but had issues measuring progress.

CONCLUSIONS

Following the results of the study, it can be concluded that each methodology has its strengths and weaknesses. As such, there is no solution for all types of projects. Various factors like the number of people in the team, how inclined to changes the requirements are or the duration of the project should be considered.

Therefore, it was observed that the majority of projects that used Waterfall are small projects, having teams of less than ten people and a duration of fewer than 6 months.

In the results of the study, it can be seen that the most popular methodology is Agile, using Scrum. Scrum reported as well the best overall satisfaction in terms of how much they enjoyed using the methodology.

In conclusion, the methodology chosen depends on each team and has to be picked specifically for that project, as no approach can satisfy all needs. The tendency is that Waterfall is used mostly by small teams for a small project that have well-defined

requirements, while Agile is more flexible and preferred when continuous feedback is important.

FUTURE WORK

Considering the small number of candidates in the research, we will try to obtain more answers in a future form. The new form will cover more metrics that can be processed. New questions will be added, choosing them after an analysis of the current data. New research should focus on differences between junior and senior developers, correlating the methodologies with the candidate generation.

REFERENCES

[1] Edward Scotcher, Rob Cole. *Brilliant Agile Project Management*, FT Press; 1 edition (January 1, 2016).

[2] Barbee Davis, *Agile Practices for Waterfall Projects*, J. Ross Publishing (October 1, 2012).

[3] Van Casteren, Wilfred. *The Waterfall Model and the Agile Methodologies: A comparison by project characteristics - short.* 10.13140/RG.2.2.10021.50403, 2017.

[4] Ken Schwaber, *Agile Project Management with Scrum*, Microsoft Press; 1 edition (February 21, 2004).

[5] Petersen K., Wohlin C., Baca D. *The Waterfall Model in Large-Scale Development*. In: Bomarius F., Oivo M., Jaring P., Abrahamsson P. (eds) Product-Focused Software Process Improvement. PROFES 2009. Lecture Notes in Business Information Processing, vol 32. Springer, Berlin, Heidelberg, 2009.

[6] Matkovic, Pedja & Tumbas, Pere. *A Comparative Overview of the Evolution of Software Development Models*. Journal of Industrial Engineering and Management. 1. 163-172. 2010.