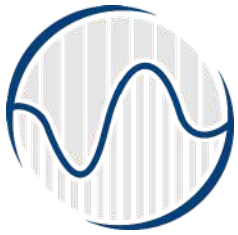
Academy of Technical and Art Applied Studies, Belgrade, Serbia
School of Electrical and Computer Engineering

Pavlov Nikola

THE APPLICATION AND COMPARISON OF AGILE METHODOLOGIES IN IT PROJECTS

- Final Thesis -



Belgrade, July 2021.

Candidate: **Nikola Pavlov**

Index number: **IS-3/18**

Study department: **Information systems**

Thesis: **THE APPLICATION AND COMPARISON OF AGILE METHODOLOGIES IN IT PROJECTS**

Basic subjects:

1. **The comparison of agile methodologies and extreme programming**
2. **The description of the Waterfall, Scrum, and Kanban methodologies**
3. **The comparison of the Waterfall, Scrum, and Kanban methodologies**

Belgrade, July 2021.

Mentor:

_____

Dr Živorad Vasić, prof.

ABSTRACT:

Basic principles of agile methodologies, as well as their practical use in real IT projects, are described in this paper. Different ways of ranking the importance of the task,their prioritization and development, as well as fixing bugs and handling technical debt, are shown through Scrum, Kanban and extreme programming methodologies. The comparison of these methodologies is described and there are examples of when it is appropriate to use each one of them.

Key words: agile methodologies, scrum, kanban, extreme programming, methodologies, IT, prioritization.

Table Of Contents:

1. INTRODUCTION

1.1. THE INTRODUCTION OF AGILE METHODOLOGIES

In project management, specifically in the management and coordination of software projects, agile methods are becoming the industry standard and are used widely. Businesses are adapting and are taking on new ways of management, and agile methods are proven to be successful.

Agile project management is becoming the new standard. Initially it was introduced to manage software projects, but its benefit has been spotted by other industries, so now a lot of companies in industries other than software and IT are adapting the new management ways - as they see it beneficial to their business.

1.2. THE HISTORY AND ORIGIN OF AGILE METHODS

At the end of the 20th century, the beginnings of what we now call agile methods appeared. In software development, iterative and incremental ways to create software projects are emerging for the first time. In software development, iterative and incremental ways to create software projects are emerging for the first time. These new ways of management appear as a reaction to some more conservative and then widely used ways, such as the waterfall method, which is much stricter and more regulated in many ways, and which has much less room for error. *The waterfall* is still used in many companies, but it is considered to have been surpassed in software and IT projects.

To shape and formalize new ways to develop software products, seventeen developers met in the U.S. state of Utah in 2001, and produced the Manifesto for Agile Software Development.

1.2.1. Manifesto for Agile Software Development

This manifesto was a document that served as the first guide to new, modern software development.

Its values are as follows:

- **Individuals** and **interactions** rather than processes and tools
- **Software that works** before extensive documentation
- **Cooperation with the client** before negotiating the contract
- **Adapting to change** rather than blindly following a plan

This does not mean that the values on the right side of the list should be completely ignored because of those on the left, but that in agile development the values on the left side take precedence.

As Scott Ambler, one of the popular software engineers and advocates of agile methods, explains:

- Tools and processes are important, but it is more important to have competent people working together effectively
- Good documentation is useful in helping people to understand how the software is built and how to use it, but the main point of development is to create software, not documentation
- A contract is important but is no substitute for working closely with customers to discover what they need
- A project plan is important, but it must not be too rigid to accommodate changes in technology or the environment, stakeholders' priorities, and people's understanding of the problem and its solution

Manifesto for Agile Software Development is based on twelve principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
4. Business people and developers must work together daily throughout the project
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation
7. Working software is the primary measure of progress
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
9. Continuous attention to technical excellence and good design enhances agility
10. Simplicity–the art of maximizing the amount of work not done–is essential
11. The best architectures, requirements, and designs emerge from self-organizing teams
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

1.2.2. The Evolution of Agile Methodologies

In the last few decades, in software and IT projects, there have been significant changes in management and management methods. Agile methodologies appear in response to the needs that have emerged in the software industry, which is relatively young compared to other industries.

Creating software is different from creating material goods, so the way you manage your business must be different.
The word "agile" by definition means fast, easy or adaptive - “able to move quickly and easily”. This adjective was taken to describe the nature of this way of management.
In software projects, problems can often occur when client requests change in the middle of a project, or when an unresolved error or similar problems occur. Then the model where the means and resources are predefined cannot function because the ultimate goal changes and that affects the course of the project and the estimation of time and material or human resources.
Working agile means being ready for these changes and adapting quickly. Software product development teams have adopted these methods and modified them to best suit the type of product being made, as well as the time and size of the teams.
Different modified agile methods emerge from different types of products, each of which has grown into its own independent way of management, with its own rules and principles.

2. PROJECT MANAGEMENT

2.1. THE PROJECT

A project is defined as a series of tasks that must be completed in order for a feat or a goal to be achieved.
According to the definition of The Project Management Institute, a project is “a temporary effort to create value through a unique product, service or result”. Depending on the complexity and difficulty of the project, it can be developed by an individual, or by hundreds of people.

2.1.1. Characteristics of a project

A project is a set of tasks that are completed and fulfilled with a common goal. Some of the features of the project are:

- **Clear start and end times of the project** - Every project must have a clearly defined beginning and end of development, as well as a clear definition of success or failure
- **The project needs to create something new** - Each project is unique, should create a new product, and should never be repeated or completely replicated
- **The project has limits** - Every project functions and develops within certain limits, whether they are financial, material, human resources or time
- **The project is not an everyday routine** - We must not confuse projects with processes that are routine and everyday, and represent a classic job in the company - they are a unique activity

2.1.1. Diversity in projects

Depending on the nature of the projects, their goals, time and resources, they can be different.
They can be big, like various Hollywood movies, or building large buildings and bridges. Such projects may require a lot of time and financial resources, as well as human resources, to be successfully developed. It can sometimes take years to build them. Smaller projects can be made in shorter periods of time and at lower cost - they can be completed in weeks or months, or even days, if we define a minor home repair or something similar as a project.
Teams on different projects can be large and diversified, but there can also be an individual who will work on the project himself.

2.1.2. Types of projects

Some examples of different project types:

- **Traditional** - These projects are usually developed through phases - start, plan, develop, monitor and end. For traditional projects, the "waterfall" model is often used, which we will talk about later. This type of development is mainly used for projects that have a large infrastructure.
- **Agile projects** - Software projects are most often developed in an agile way. They are adaptable and their iterations usually last shorter. They are focused on people and how the team works together.
- **Projects with distributed teams** - By this we mean teams that are not in the same place. Here, project development does not depend on location, so different teams are often in different places, and often individuals within one team do not work from the same place. An

example of this type of management is the increasing number of freelance projects.

- **Agency projects** - This type defines projects that are made by an agency that is employed as a contractor, which means that the project is made by an external agency and not by the company that owns it. These types of projects are most often in the marketing or design fields.

2.1.3. Project development lifecycle

Projects are often divided into five stages of development. Each of these phases must have its own time limit and its own tasks.
These phases serve to keep teams focused on project design.

They are:

1. **Starting a project** - The project is officially started and named. A broader plan is made and goals, risks and participants in the project are defined.
2. **Planning** - A plan of clearly defined activities is made, deadlines are set and the necessary resources are allocated. Tasks are narrowed down to smaller activities that are easier to manage.
3. **Development** - The previously defined project plan enters into force and is implemented. At this stage, the team is working to meet the requirements.
4. **Monitoring** - Monitoring and quality control of the project happen along with development. At this stage, the manager must ensure that the teams work efficiently and succeed in meeting the requirements. Deadlines, resources, and risks are taken into account.
5. **Conclusion** - The project is coming to an end, and the project managers are in charge of handing over all the necessary documents to the client in addition to the project itself so that the project can be officially complete. In this phase, the team discusses together the course of the project, what the members have learned and how they could have done the project better.

When the project is completed on time and within the planned financial resources and costs and has met all the client's requirements, it can be considered successful.

3. AGILE METHODOLOGIES

3.1. POPULAR AGILE METHODS IN IT PROJECTS

As new projects emerge, their management methods are constantly being adapted and improved. Over the years, various modifications of these management methods have emerged. Each of the different methods has its reasons, its rules and the ways in which it is performed. Teams follow the rules that dictate these methods in order to be as efficient and better as possible during the project development.

3.1.1. Scrum

Scrum can be defined as an iterative and incremental framework that mostly smaller teams use - around 5 to 15 members. As a management method it's defined somewhere around 1995. but it started gaining popularity after 2001. when the *Agile Alliance* is formed. Scrum framework is created around a set of roles, rules, elements and events.

ROLES OF A SCRUM TEAM

The Scrum framework defines only three roles in the team - the role of *product owner*, the role of *scrum master*, and the role of *team members*.

Product owner controls the *backlog* and defines the requirements.

The Scrum Master makes sure that the Scrum rules are followed and that obstacles are removed so that the team can work smoothly to meet the requirements.

Team members are programmers, developers, designers and other roles that work on requirements, new features, fixing bugs and designing the software.

All members of the Scrum team work together, and even though the Product owner and Scrum master coordinate the requirements and the general framework of the work being done and monitored, all members are equally important for the success of the project. This is one of the agile principles and a rule of Scrum.

SCRUM EVENTS

There are four main events that take place during the development of a Scrum-led project - *sprint planning*, *daily standup*, *sprint review,* and *sprint retrospective*. These events are time-limited so as not to take up much of the valuable time during the day, and serve to facilitate processes as quickly as possible and to have a constant flow of information between all team members and stakeholders who are out of the team. They are called **events**, but we can look at them as **team meetings**.

Sprint planning - The following sprint is planned at this meeting. For this sprint, previously analyzed tasks are determined and they are assigned to the team members who will perform them. In the context of Scrum teams, tasks are often called stories or "tickets", and this comes from the numerous project management software solutions that named the tasks in their programming environment like that because tasks visually look like cards or "tickets" with information.
Daily standup - This is a meeting that takes place every day and as a rule it must not last longer than 15 minutes. It usually happens at the beginning of working hours. This meeting goes through the tasks of the current sprint and all team members present the progress of the tickets they worked on. Everyone declares if they have finished the ticket or there have been some

blocks, and the product owner and scrum master take this into account in order to solve these problems. If the problem is bigger, it is usually not resolved on the spot, but separate meetings are scheduled later, in order to comply with the 15-minute rule. The word “standup” symbolizes the quick nature of the meeting, imagined to be held in an office while everyone is standing - meaning that it will not last long.

Sprint review - The sprint review refers to the meeting that takes place after the end of the current sprint. In this meeting it’s demonstrated what has been achieved during the sprint, these achievements are discussed with stakeholders and the product backlog is changed depending on the requirements.

Sprint retrospective - Like the sprint review, the retrospective takes place at the end of the current sprint, but it aims to give the Scrum team a look at the problems and challenges they have encountered. Team members then work together to solve these problems and come up with potential solutions. The usual questions:

- What went well during the sprint?
- What went badly?
- Which steps can we take to improve the processes?

A retrospective serves to solve such problems.

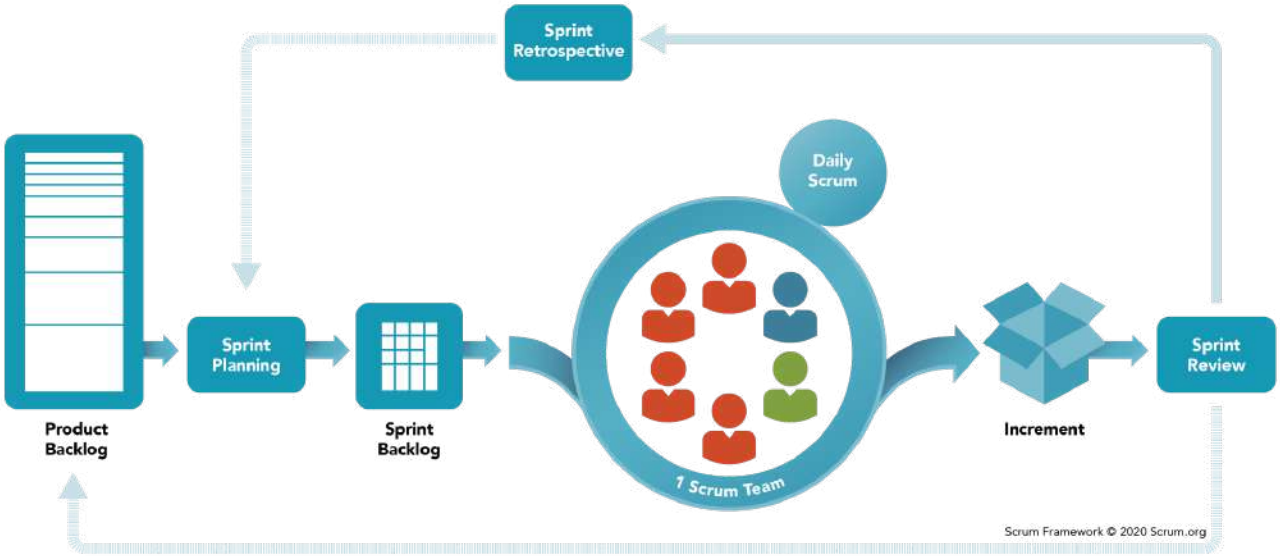
SCRUM ELEMENTS

Besides events and members, Scrum is also defined by its elements. Most important elements are *sprint*, *product backlog*, *sprint backlog* and *product increment*.

Sprint - Represents a timeframe of 2 to 4 weeks for which tasks are completed by team members. During this period, daily standup meetings are held every day.

Prior to the start of each sprint, prioritization is performed and the tasks for the next sprint are determined. At the end of the sprint, review and retrospective are held so the stakeholders are updated about the product and possible errors are brought up.

Product backlog - It contains the requirements to be met in one of the future sprints. In practice, we can see it as a long list of tickets that need to be prioritized in order to enter one of the following sprints.
Sprint backlog - It contains the requirements that team members are currently working on, i.e. the tickets that are in the current sprint.
Product increment - It represents the result of the work spent during the sprint, i.e. the increment of product requirements.



Sl. 3.1. Lifecycle of a Scrum project
Source: Scrum.org website

The image above shows the iterative process of a sprint. Tasks are extracted from the *product backlog* and inserted into the next sprint on *sprint planning*. During the sprint, tasks are done, followed by incrementation and sprint review. After that retrospective, the process starts all over again.

3.1.2. Kanban

Kanban is a method of management that defines, manages and improves the way we work in the information business. It helps make the job easier to visualize.
This management framework began in the production processes, but later it also became one of the agile management methods.
The Japanese word "Kanban" can be translated as a visual board or sign, and has been used as a concept in production management systems since the 1960s.

It is believed that the beginnings of the Kanban method originated in Toyota in the 1940s, when the company announced a new way of time management, "just in time". This essentially meant that the product was made in relation to demand, and not as the practice that was the standard back then - to make it first and then push it to market.

THE KANBAN METHOD

At the beginning of the 21st century, the leading companies discover Kanban as a tool that will change product delivery for the better.
Focusing on efficiency, Kanban is slowly transitioning from the automobile industry to other industries.
The “Kanban method” we know today officially arrives in 2007.

Getting started using the Kanban method can be very simple - o begin with, it is enough to set up a board with three basic columns - “**To do**”, “**In progress**”, and “**Done**”. These three columns indicate tasks and serve to monitor work in real-time.

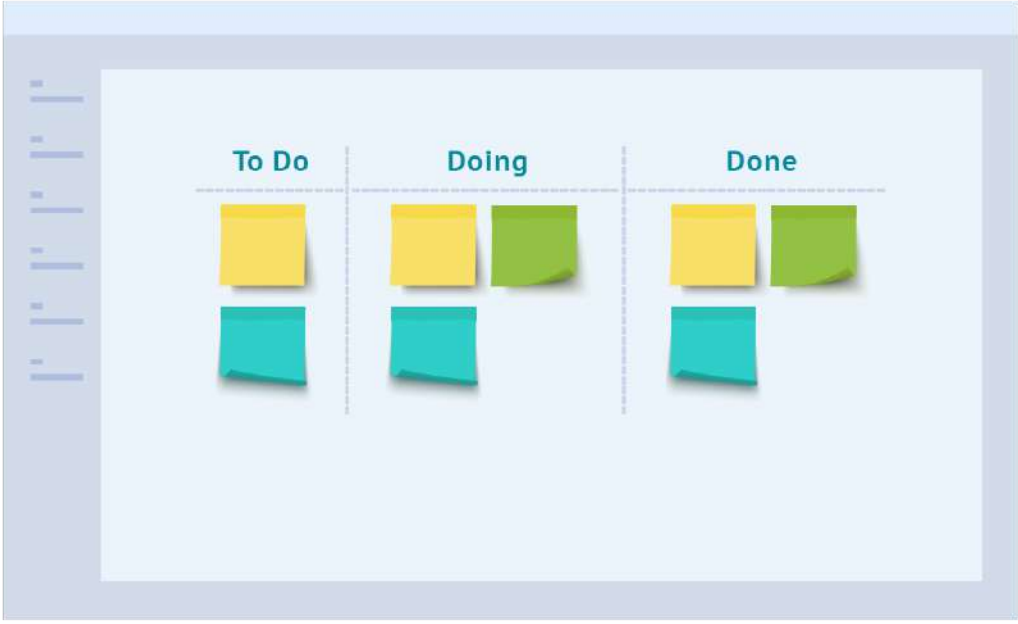
KANBAN PRINCIPLES

When we want to implement the Kanban method as a management framework for our project, we must follow some principles. There are 6 basic principles that should be present in order to successfully implement the Kanban method.

1. **Visualization of the work process** - In order to be able to visualize the processes within the Kanban system, we need a board, cards, and columns. Earlier we mentioned "tickets" - we use cards in the same context. Each card represents the subject of work, that is, the requirement of the product that is currently being worked on.
   When work is started on a card, it is moved from the "To Do" column to the "Working" column, and then other team members see the state of the work and know not to take on the same task. Thus, all members can take care of the process. When the work on the card is finished, it is dragged into the "Finished" column.
2. **Limit the number of cards in use** - In order to have an easier view of the entire work process, we need to limit the number of cards for each column. For example, we can accumulate a large number of cards that are currently being worked on. This is bad because it makes it harder for us to prioritize what needs to be completed first. When we limit the number of cards, we will not start work on new cards if enough old ones have been completed.
3. **Manage the flow** - This means that we must clearly define processes and definitions of completed cards and their states. In order for team members to follow the processes, they must be clearly defined.
4. **Make policies explicit** - By having an explicit understanding of issues, operations, and rules, discussions become more rational and objective. These need to be documented and shared across the project team. The intention is to prevent emotion and subjective views from seeping into the decision process.
5. **Implement feedback loops** - As in Scrum, we need to receive feedback from both team members and clients at regular intervals.

This is one of the things that makes the Kanban method one of the agile methods.

6. **Improve collaboratively, and evolve experimentally** - In Kanban, collaboration and experimentation go hand in hand as long as there is clarity and consensus on how to approach work and issues.



Sl. 3.2. Primer jednostavne Kanban table

3.1.3. Extreme programming (XP)

Extreme programming is an agile method that focuses on producing high-quality software.

XP is the most narrowly defined agile method in the sense that its principles are clear and must be respected, unlike some other agile methods that are subject to change and modification depending on the team that implements them.

USE OF XP

To use extreme programming and choose it as a project management method, it is necessary that the project has certain characteristics.

Some of the project characteristics:

- Dynamically changing software requirements
- There are risks arising from tight deadlines and the use of new technologies
- A small team of co-located developers
- It is necessary that the technology used on the project allows the use of automatic functional and unit tests

These features are listed at www.extremeprogramming.org and were designed by the proponent and one of the creators of extreme programming, Don Wells.

Because of the strict set of rules that must be followed, most projects do not use extreme programming, but some variation of it.
Regardless, proponents of this agile method believe that it is certainly good to adhere to its principles as best as possible.

THE VALUES OF EXTREME PROGRAMMING

The extreme programming method propagates some of the following values:

- **Communication** - Software development can be seen as a team sport, and XP focuses on transferring knowledge from one team member to everyone else. A good flow of information means easier prevention of problems and greater success.
- **Simplicity** - Under this, XP asks what is the simplest thing that works? The point of this is for the team to focus on the things they do, and to reduce "waste" - unnecessary code that can cause problems later.
- **Feedback** - Through constant feedback, the team is always aware of things that need to be fixed and improved.
- **Courage** - Through this value, XP emphasizes self-initiative, or in a more extreme case, action in the face of fear. It takes courage to stop doing something that's wrong and start doing something that's good for the team - essentially the courage to accept a mistake.
- **Respect** - In order for communication to be better and feedback to be accepted more easily, team members must have respect for each other.

BEST PRACTICES IN XP

Extreme programming practices have changed since this method was first conceived. These are the 12 original practices:

- Planning
- Fewer software deliveries
- Metaphors (program vision)
- Simple design
- Testing
- Refactoring
- Pair programming
- Joint ownership of the project
- Continuous integration
- 40-hour weekly working hours
- Client on site
- Code standards

The main contribution made by the extreme programming method is a set of programming and engineering rules and standards that teams can use to produce more effective and better code.

A lot of engineering teams start out with some other agile method, but then see the need for more disciplined engineering practices, and start applying extreme programming standards and values.

Another contribution of XP is the practice of writing excellent code. Extreme programming puts the code itself and the engineering quality of the product and project at the center.

4. THE WATERFALL METHOD

4.1. THE INTRODUCTION OF THE WATERFALL METHOD

The waterfall method represents a linear approach to project management, according to which, at the beginning of the project, requirements are collected from the client and stakeholders, and then a project plan is developed that develops those requirements through sequences.

This is a well-structured method that has been around for many years because it has proven to be successful. It can also be said that it is one of the oldest methods and one of the first approaches to project management.

Some of the industries where the waterfall method is used on a daily basis include the construction industry, IT and software development. However, the term "waterfall" has recently been used most often in the sense of software development.

4.2. METHODS AND PHASES OF MANAGEMENT

The waterfall model is divided into several different stages that take place sequentially.
In order for the next phase to begin, the previous one must be completed.
Those phases are:

1. **Requirements gathering** - One of the key aspects of the waterfall method is that all user requirements are collected at the beginning of the project, which allows each next phase to be planned without additional changes and without client involvement until the product itself is finished. It is assumed that all requirements can be collected at this stage.
2. **System Design** - The design phase can be further divided into two sub-phases - logic design and physical design. The logic design

subphase is possible when theories about solutions are created, while the physical design subphase means that those theoretical ideas and schematics are turned into concrete specifications. Then the infrastructure of the system is designed.

3. **Implementation** - In this phase, developers, based on previous requirements, specifications and designs, create concrete code. This represents software development along with writing and running tests.
4. **Confirmation by the client** - In this phase, the client reviews the product to ensure that it meets the previously defined requirements at the beginning of the project. This is done by showing the client a fully finished and functional product.
5. **Monitoring** - The client or user regularly uses the product and reports all errors, and the development team solves them until the user is satisfied with the complete product.

4.3. THE ARGUMENTS FOR THE WATERFALL METHOD IN SOFTWARE PROJECTS

Since the beginning of the 21st century, agile methods are becoming more and more popular every day. Today, companies developing new products and emerging startups often choose Scrum as a management methodology without any consideration of other options, as it is considered the standard in IT projects.
However, there are many arguments for using the waterfall method.

For example, time spent on clearly defining requirements and potential problems in the early stages of a project can yield significant savings in later stages. A problem found early on is much cheaper to fix than the same problem later in the project.

As an example, we can take software development using a selected software tool. With an agile approach, we don't necessarily know what the final product will look like. Towards the end of development, it turns out that we need some new functionality that the current tools do not provide, and therefore we have to look for other tools and go back a few steps.

If we had known at the beginning that we were going to implement this functionality, we would have chosen different tools to work with.

Also, the waterfall method places great importance on documentation. Documentation is essential for knowledge sharing, because if one of the team members leaves the project before the product is ready, there may be problems with transferring knowledge to new members. That is why documentation is an important aspect of project management.

The waterfall model provides a linear approach that is easy to follow through stages and processes.

5. COMPARISON OF AGILE METHODS AND THE WATERFALL METHOD

In order to gain a deeper understanding of the different methodologies, their meaning and use, we need to compare them.

We will compare the mentioned methods in terms of efficiency, speed of development, ability to accept and adapt to errors, and tools used for their implementation.

5.1. AGILE METHODOLOGIES

5.1.1. Scrum - Key features

In modern IT projects and teams of approximately 5 to 15 people, Scrum is often chosen as the default option for management.

What distinguishes Scrum from other methods is that it focuses mostly on the team itself and on improving the process so that team members work together as well as possible. The principles of teamwork and knowledge sharing are followed.

Scrum is not adequate for projects done by one person, because too many processes can slow down progress, and there is no point in completing them if a person is working on the project alone. It is also not adequate for large teams, because it takes care of what exactly is being worked on, and a loss of control could easily occur.

Scrum is good for projects that are adaptable and error-prone - we often see this in IT projects where requirements change easily, for example in relation to the market. Software projects are good because changes can be made quickly and immediately tested in the market with users.

A good example of a real non-software project where Scrum would be a much more adequate solution than the waterfall method:

- My father works in a company that installs elevators. Currently, the workers come to the field, wait for the material to arrive, and when the material and tools arrive, they start working. Scrum can be a potential solution here because building elevators is similar to building software - there is no mold, but each elevator is different in relation to the building, space or design. It often happens that the wrong parts are delivered, and then the work slows down. We can solve this through sprints, where one sprint will focus on delivering the necessary tools and materials, and the next one will not start at the same time or until this one is finished. When this sprint has been verified to be successful, the next sprint can begin, i.e. the installation of the elevator. If the sprint has failed, the installation of the elevator must not begin, because working with the wrong parts will only slow down the project.

Agile methodologies should also not be too strictly separated, as their principles can overlap and be used together depending on the needs of the project.

The rhythm of the Scrum method is shown through regular two-week sprints, the product is released in fractions at the end of each sprint, the measurement of success is done through various points such as speed or work consumed.

The main roles of the Scrum team are the product owner, scrum master and other team members, mostly developers or designers.

The philosophy behind making changes is that requirements should not change during the current sprint.

5.1.2. Kanban - Key features

Kanban is an agile methodology that basically does not have so many processes, but mostly focuses on the visualization of work and business requirements.
Kanban is a good option when the team does not have a lot of time for meetings, the work process is continuous and uniform, the team wants to limit meetings and planning in order to focus on delivering the product.

It is also adequate when new parts of the product are continuously delivered as opposed to delivered in fixed iterations as is the case with the Scrum method.

Kanban can also be used as a visualization tool to help small teams, where even two or three people can coordinate more easily using a Kanban board.

Unlike the Scrum framework, the rhythm of the Kanban method is uniform and constant. Deliveries are made constantly.

There are no clearly defined roles in the team.

The tickets on the board must be limited to a pre-agreed number so that the board does not become messy and unclear, and to make it easier to clearly see who is working on what and how many tasks are in circulation.

The philosophy of embracing change is that change can happen at any stage of development.

5.1.3. Extreme Programming (XP) - Key features

Extreme programming is an agile method that focuses on processes, quality software and code standards. In order to implement this method correctly, a team of experienced programmers is generally required.

XP is a satisfactory option when the requirements and the product are expected to experience major changes every few months.
It is usually used in smaller teams, from 2 to 12 people, but in some cases larger teams have also seen success.

It is important that all code is tested and that it is functional, that is, of good quality.

Unlike the Scrum method where sprints last 2-4 weeks, XP has a rhythm of iterations lasting 1-2 weeks. Smaller iterations allow for easier testing and faster development.

Scrum does not allow changing requirements during a sprint, but XP allows this.

While Scrum focuses on teamwork and Kanban focuses on visualization, XP focuses on good engineering practices.

The XP method is known for its ready-to-apply approach, which means that the team should always be ready to apply the means to get the tasks done in the best possible way.

5.2. THE WATERFALL METHOD AND AGILE METHODOLOGIES

Agile methodologies differ from each other, but are guided by some similar principles, while the waterfall method is completely different.

These two methods are driven by completely different project management philosophies.

In agile methods, the creation of a product will go through continuous iterations - as an example we can see the iteration of product development, review, testing, approval, official code delivery. This is one of the iterations that a task goes through. These iterations are constantly repeated. We can see the waterfall as processes from these iterations, only that it does not develop through multiple iterations but rather one big set of several long phases.

Following this method is simpler, and involves going through each of these stages in detail. Everything must be clearly planned at the beginning and there is no room for adaptation, but the originally defined product is created.

Some key differences:

- Agile methods are iterative and incremental, while waterfall is linear
- Agile methods divide the project into sprints, waterfall divides it into phases
- Agile methods help to build multiple small projects, waterfall focuses on building one large project
- Agile methods focus on client satisfaction, waterfall on successful project delivery
- Agile methods allow for changes at any time, waterfall does not
- Testing is done constantly in an agile approach, while in waterfall it is done at the end

6. CONCLUSION

The aim of this paper was to describe agile methodologies - Scrum, Kanban and Extreme Programming; as well as the waterfall method, to make a comparison between them and to show in which projects it is adequate to use which methodology and why.

One of the key differences between agile methodologies and traditional ones is that agile methodologies have an iterative and incremental approach, and adapt to changes, while more traditional methodologies like waterfall are more clearly defined from the start.

Integrating agile methodologies into an IT project can have a positive impact on productivity, but only if we clearly define the criteria by which we measure success, and only when we look at all the facts. Depending on various factors like time, cost, team size and software quality, we can choose the right method for the project.

The main factor when choosing one of these methods should be the productivity of the team, because it will contribute to the greater success of the project itself.

7. INDEX OF TERMS

8. LITERATURE

[1] Jeff Sutherland, J. J. Sutherland, “Scrum: The Art of Doing Twice the Work in Half the Time”

[2] Eric Ries, “Lean Startup”

[3] Eric Ries, “The Startup Way”

[4] Website “http://www.extremeprogramming.org/”

[5] Website “https://www.scrumalliance.org/”

[6] Website “https://www.scrum.org/”

[7] Website “https://www.agilealliance.org/”

[8] Website “https://agilemanifesto.org/”

DECLARATION OF ACADEMIC INTEGRITY

Student(first name, name of one parent, last name: __Nikola, Nenad, Pavlov___
Index: _____IS-3/18_____

Under full moral, material, disciplinary and criminal responsibility, I declare that the final work, entitled:

_____THE APPLICATION AND COMPARISON OF AGILE _____
_____METHODOLOGIES IN IT PROJECTS_____

1. The result of own research work;
2. That I did not submit this work, neither in whole nor in parts, to other higher education institutions;
3. That I have not violated the copyrights, nor misused the intellectual property of other persons;
4. That I indicated or cited the work and opinions of other authors that I used in this paper in accordance with the Instructions
5. That all works and opinions of other authors are listed in the list of literature/references that is an integral part of this work, listed in accordance with the Instructions;
6. I am aware that plagiarism is the use of someone else's work in any form (such as quotations, paraphrases, images, tables, diagrams, designs, plans, photographs, films, music, formulas, websites, computer programs, etc.) without crediting the author or presenting other people's author's works as mine, punishable by law (Act on Copyright and Related Rights) as well as other laws and corresponding acts of the School of Electrical and Computer Engineering in Belgrade;
7. That the electronic version of this work is identical to the printed copy of this work and that I agree to its publication under the conditions prescribed by the acts of the School of Electrical and Computer Engineering in Belgrade
8. That I am aware of the consequences if it is proven that this work is plagiarized

Belgrade, __, __, 201_. godine Handwritten signature of the student

_____