

UNIVERSITY OF LJUBLJANA
SCHOOL OF ECONOMICS AND BUSINESS

MASTER'S THESIS

**A COMPARISON BETWEEN SCRUM AND KANBAN METHODS IN
IT PROJECTS**

Ljubljana, September 2025

MILICA SPASIĆ STEVANOVIĆ

AUTHORSHIP STATEMENT

The undersigned Milica Spasić Stevanović, a student at the University of Ljubljana, School of Economics and Business, (hereafter: UL SEB), author of this written final work of studies with the title **A comparison between Scrum and Kanban methods in IT projects**, prepared in collaboration with mentor Prof. dr. Talib Damij,

DECLARE

1. this written final work of studies to be based on the results of my own research;
2. the printed form of this written final work of studies to be identical to its electronic form;
3. the text of this written final work of studies to be language-edited and technically in adherence with the UL SEB Technical Guidelines for Written Works, which means that I cited and / or quoted works and opinions of other authors in this written final work of studies in accordance with the UL SEB Technical Guidelines for Written Works;
4. to be aware of the fact that plagiarism (in written or graphical form) is a criminal offence and can be prosecuted in accordance with the Criminal Code of the Republic of Slovenia;
5. to be aware of the consequences a proven plagiarism charge based on the this written final work could have for my status at the UL SEB in accordance with the relevant UL SEB Rules;
6. to have obtained all the necessary permits to use the data and works of other authors which are (in written or graphical form) referred to in this written final work of studies and to have clearly marked them;
7. to have acted in accordance with ethical principles during the preparation of this written final work of studies and to have, where necessary, obtained permission of the Ethics Committee;
8. my consent to use the electronic form of this written final work of studies for the detection of content similarity with other written works, using similarity detection software that is connected with the UL SEB Study Information System;
9. to transfer to the University of Ljubljana free of charge, non-exclusively, geographically and time-wise unlimited the right of saving this written final work of studies in the electronic form, the right of its reproduction, as well as the right of making this written final work of studies available to the public on the World Wide Web via the Repository of the University of Ljubljana;
10. my consent to publication of my personal data that are included in this written final work of studies and in this declaration, when this written final work of studies is published;
11. that I have verified the authenticity of the information derived from the records using artificial intelligence tools.

Ljubljana, August 15th, 2025

Author's signature: *M. Spasić-Stevanović*

TABLE OF CONTENTS

1 INTRODUCTION	1
2 TRADITIONAL PROJECT MANAGEMENT	2
2.1 Project management	3
2.2 Project Management Life Cycle	6
2.3 IT Project Management.....	7
2.3.1 Waterfall	7
2.3.2 Spiral.....	8
2.3.3 V model	9
2.3.4 RAD.....	9
3 AGILE PROJECT MANAGEMENT.....	10
3.1 Defining Agile	10
3.1.1 Incremental development	12
3.1.2 Iterative development	13
3.2 Comparative Analysis of Agile vs. Traditional Methodologies.....	14
3.3 Advantages of Agile.....	15
3.4 Disadvantages of Agile	15
4 SCRUM.....	16
4.1 Defining Scrum roles.....	17
4.1.1 Product Owner.....	17
4.1.2 Scrum Master.....	18
4.1.3 Development Team.....	19
4.2 Defining Scrum activities.....	19
4.2.1 The Sprint	19
4.2.2 Sprint Planning	20
4.2.3 Daily Scrum.....	21
4.2.4 Sprint Review	21
4.2.5 Sprint Retrospective	21
4.3 Defining Scrum artifacts.....	22
4.3.1 Product Backlog	22
4.3.2 Sprint Backlog	23
4.3.3 Product Increment.....	23

4.4 Advantages and disadvantages of Scrum framework	24
5 KANBAN	25
5.1 Kanban framework overview	25
5.2 Kanban principles	27
5.3 Visualize workflow	27
5.4 WIP limits	29
5.4.1 Little's law	29
5.4.2 Different approaches in setting the WIP limit	30
5.5 Manage flow	30
5.6 Making process policies explicit	32
5.7 Implementing fast feedback loops	32
5.8 Drive improvements based on methods and models	32
5.9 Advantages and disadvantages of Kanban framework	32
6 SCRUM AND KANBAN IN PRACTICE	33
6.1 Project description	33
6.2 Using Scrum	34
6.3. Using Kanban	36
7 SCRUM AND KANBAN COMPARISON	38
7.1 Task flow and delivery rhythm	38
7.2 Work in progress (WIP) and workflow management	38
7.3 Flexibility	38
7.4 Team collaboration and coordination	39
7.5 Team motivation	39
7.6 Challenges and limitations	39
7.7 Summary	39
8 CONCLUSION	40
REFERENCE LIST	41

LIST OF APPENDICIES

Appendix 1: Summary in Slovene Language	1
Appendix 2: Scrum progress tracking	2
Appendix 3: Kanban progress tracking	6

LIST OF TABLES

Table 1: Scrum and Kanban Comparison.....	40
---	----

LIST OF FIGURES

Figure 1: Six-constraint Model.....	5
Figure 2: Basic Project Management Processes	6
Figure 3: The Waterfall Model	8
Figure 4: The Spiral Model	8
Figure 5: V-Model in Software Development.....	9
Figure 6: Phases of Rapid Application Development	10
Figure 7: Iterative and Incremental Development.....	13
Figure 8: Stacey Matrix adapted to Software Development.....	14
Figure 9: Scrum Events and Meetings.....	19
Figure 10: Burndown Chart Example.....	23
Figure 11: The Kanban Board	28
Figure 12: Cumulative Flow Diagram (CFD)	31
Figure 13: Scrum Progress Tracking - Sprint 1	35
Figure 14: Kanban Progress Tracking - Day 1	36
Figure 15: Kanban Progress Tracking - Day 2	37

LIST OF ABBREVIATIONS

CFD - Cumulative Flow Diagram
CPM - Critical Path Method
FDD - Feature-Driven Development
IPMA - International Project Management Association
IT - Information Technology
OOPSLA - Object-Oriented Programming, Systems, Languages & Applications
PERT - Program Evaluation Review Technique
PMBOK - Project Management Body of Knowledge
PMI - Project Management Institute
QA - Quality Assurance
RAD - Rapid Application Development
SDLC - System Development Life Cycle
WIP - Work-In-Progress
XP - Extreme Programming

1 INTRODUCTION

Agile methods have appeared as a more flexible alternative to the traditional methods of managing software development projects. These methods and their processes quickly became valuable in the IT industry, known for the rapid changes. The agile process supports the software development life cycle starting with requirements gathering, analysis, design, coding, testing and delivery of a partially developed software, which evolves with customer feedback. There are several methods through which agile projects can be implemented (Sharma et al., 2012). Most widely known, Scrum and Kanban methods, track and manage the progress of software development. These methods are flexible and can be adjusted to different team structures, different workflows, and various industries. However, using the wrong approach can result in wasteful and even harmful development processes, impacting the organization and causing inefficient development (Alqudah, & Razali, 2017). While many companies and managers have accepted agile methods and are happy to implement them, the problem occurs when they rely on the wrong framework. For example, Scrum, one of the earliest agile approaches, is still the most popular and most widely used agile framework, according to the 16th State of Agile Report (Digital, 2022). It was designed to be simple and easy to adopt (Schwaber, & Beedle, 2001), while emphasizing the creation of business value through frequent customer interaction and quick adaptation to business dynamics (Conboy, 2009). Despite being the most widely accepted, Scrum may not be the right fit for all. In some cases, a more flexible method like Kanban might be better, especially if the focus is on optimizing the workflow and improving efficiency over time.

The overall purpose of this master thesis is:

- To gain thorough understanding and deeper knowledge in project management and agile methods
- To learn about and summarize advantages and disadvantages of the Scrum and Kanban frameworks
- To determine which framework suits the best and provides better results in given circumstances

This further leads us to the initial hypothesis of this master's thesis that although Scrum is currently the most widely used agile method, it's not always the most effective one in practice. Hence, the goal is to implement and test both Scrum and Kanban frameworks within the selected company, which works on apps and software development. Additionally, a comparative analysis is going to be conducted based on the outcomes of both methods. Furthermore, the goal is to establish guidelines and provide recommendations for better usage of agile methods in IT project management.

The thesis is structured into two main parts: theoretical part and empirical part. The theoretical part begins with a description of the evolution from traditional project management to project management in the IT industry, leading to agile methods. Going

forward, the focus is on a detailed exploration of the Scrum and Kanban frameworks. Various criteria are used to analyse and compare the two methods, including their principles, board design, team roles, and specific rules that define each approach. Scrum is a project management method designed to manage frequently changing project requirements by improving communication between the project owners, developers and other team members. Scrum framework concentrates on how the team members should function in order to produce the system flexibly in a constantly changing environment (Abrahamsson et al., 2002). Kanban is another project management method for software development that emphasizes “just-in-time” delivery. The main focus of Kanban is to accurately state what work needs to be done, and when it needs to be done. It does so by prioritizing tasks, and defining workflow as well as lead-time to delivery. The Kanban process precisely presents the most important tasks that need the most attention in order to reduce the risk of incompleteness, and also to increase flexibility amongst other tasks in the project (Lei et al., 2017).

The empirical part, presented in chapters six and seven, focuses on the implementation and analysis of both Scrum and Kanban methods within a specific company project. It includes an evaluation of these frameworks based on their performance and how suitable they are for the specific project in the real world. Finally, the conclusion summarizes the key findings and provides recommendations for selecting and using agile frameworks more effectively in practice.

2 TRADITIONAL PROJECT MANAGEMENT

Project management in its basic shape and form can be traced back to centuries before it was formally defined as what we know it today. Large-scale endeavours such as the construction of the Pyramids of Giza, Great Wall of China, and Coliseum are all good examples of such projects. Despite the common assumption that these projects had unlimited resources such as workforce, budget and time, Kozak-Holland (2011) demonstrates that historical projects encountered challenges similar to modern projects, characterized by significant constraints such as scope, schedule, budget, quality, resources, and risk management. This suggested that ancient project leaders were some kind of a precursor to modern project managers, overseeing different phases, from initiation and planning through execution, monitoring, and closure. They managed large teams over extended periods, ensured continuous resource availability, maintained schedules, and met leadership expectations for the final outcome (Seymour, & Hussein, 2014).

Although some forms of project management were already noticed back in ancient societies, it is generally considered that practice of modern project management began in the 20th century. One of the first tools related to project management used for visually describing, planning and tracking project activities, breaking them into smaller chunks and showing their dependencies, was the Gantt chart, named after Henry L. Gantt (Trojanowska, &

Dostatni, 2017). In 1950's, Jim Kelley and Morgan Walker developed the Critical Path Method (CPM), a technique used for project planning and scheduling. Their goal was to reduce costs that were caused by inefficient planning, estimating and scheduling at DuPont and find "a better way to do things" (Kelley et al., 1989). Another technique for project planning that emerged around the same period was Program Evaluation Review Technique (PERT). It was developed by the US Navy as the method for tracking and managing numerous contractors working on a complex POLARIS missile project, focused on reaching project milestones (Stretton, 2023).

The formalization and institutionalization of project management as a distinct discipline began with the creation of the International Project Management Association (IPMA) in 1965. The field further advanced with the founding of the Project Management Institute (PMI) in the U.S. in 1969, a professional organization dedicated to promoting and standardizing project management practices (Seymour, & Hussein, 2014). During this period, project management remained primarily associated with the construction, defence, and aerospace industries. However, in the 1970s and 1980s, largely due to the efforts of PMI's Project Management Body of Knowledge, project management expanded and evolved into a more structured discipline and methodology as we know it today (Stretton, 2023).

2.1 Project management

Prior to giving a definition of the term project management, it is important to get a clear understanding of what the project is. The Project Management Institute (2008), in the PMBOK® Guide, defines a project as "*a temporary endeavour undertaken to produce a unique product, service, or result*". Various authors provide additional definitions further elaborating the project definition. For instance, Heagney (2011) states that a project is characterized by a clear beginning and end (time), a budget (cost), a defined scope or scale of work, and specific performance requirements.

Lock (2014) categorizes projects into four primary types:

1. Civil engineering, construction, petrochemical, mining, and quarrying projects
2. Manufacturing projects
3. Management and business change projects
4. Scientific research projects

Expanding on these characteristics, Kerzner (2009) defines a project as any set of tasks and activities that:

- Aim to achieve a specific objective within given requirements,
- Operates within the specified start and end dates,
- May be subject to funding constraints,
- Require human and nonhuman resources, such as funds, personnel, and equipment, and

- Involve multiple functions, often spanning across various departments or functional areas.

The Project Management Institute (2008) further defines project management in the PMBOK® Guide as the "*application of knowledge, skills, tools, and techniques to project activities to meet the project requirements*", which is considered as a widely recognized definition. According to Lock (2014), the goal of project management is to efficiently plan, organize, and control project activities to achieve successful completion despite the risks and other challenges. In addition to previous definitions, Kerzner (2009) adds that project management involves "the planning, organizing, directing, and controlling of company resources for a relatively short-term objective aimed at completing specific goals."

Traditional project management, that is often termed as the "waterfall" methodology, is characterized by its sequential, linear progression, in which each phase is completed before the next upcoming phase begins. This approach expects strict following of the timelines and budgets, predictable project conditions and clearly defined requirements from the start. The emphasis is placed on thorough planning and control throughout each stage of the project lifecycle (Ciric et al., 2019).

As Salameh (2014) notes, project management focuses on meeting the demands of time, scope, budget and quality while adhering to stakeholder requirements. This is achieved through the framework of 10 essential knowledge areas defined in the PMBOK® Guide by the Project Management Institute (2013):

1. Project integration management - ensures cohesive planning and execution;
2. Project scope management - defines project boundaries and controls changes;
3. Project time management - focuses on scheduling;
4. Project cost management - covers budgeting and financial tracking;
5. Project quality management - includes assurance and control;
6. Project human resources management - manages the project workforce;
7. Project communications management - involves information sharing with stakeholders;
8. Project risk management - addresses identifying and mitigating potential issues;
9. Project procurement management - handles obtaining necessary goods and services;
10. Project stakeholder management - includes identifying stakeholders, making them involved throughout the project and aligning their expectations.

Together, all of these areas provide a clear structure for establishing processes and managing projects effectively.

Once these foundational elements are in place, project management focuses on optimizing resources to meet defined objectives, while respecting three constraints:

- Time - represents the schedule and the deadline by which each phase of the project, as well as the entire project, should be completed;

- Cost - stands for the budget allocated to the project, covering all the expenses (labour, resources, equipment);
- Scope - corresponds with the amount of work required including the specific objectives and performance requirements that must be achieved.

In project management, the constraints of scope, time, and costs are commonly known as the "Triple Constraints," serving as foundational elements that collectively define and influence project's limitations and success criteria (Schwalbe, 2017). Heagney (2011) highlights the relationship among these constraints through the formula:

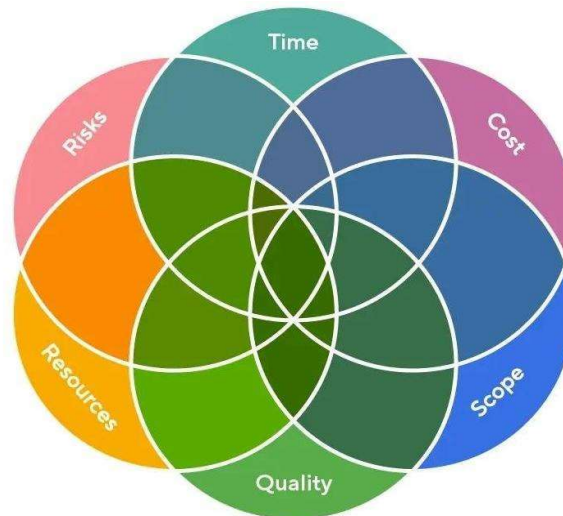
$$C = f(P, T, S) \quad (1)$$

This expression (1) indicates that Cost depends on Performance, Time, and Scope and since each constraint is interdependent, this means that changes in one can directly impact the others.

Recognizing the increasing complexity of modern projects, the PMBOK® Guide added additional constraints over time (see Figure 1):

- Quality - means ensuring the required standard the project deliverables must meet;
- Resources - refers to people, materials, tools and assets needed to finalize project tasks;
- Risk - involves potential challenges or uncertainties that could impact project success.

Figure 1: Six-constraint Model



Source: Eby (2022).

The need for a multidimensional approach and balancing alongside previously mentioned set of constraints are crucial for maintaining project success.

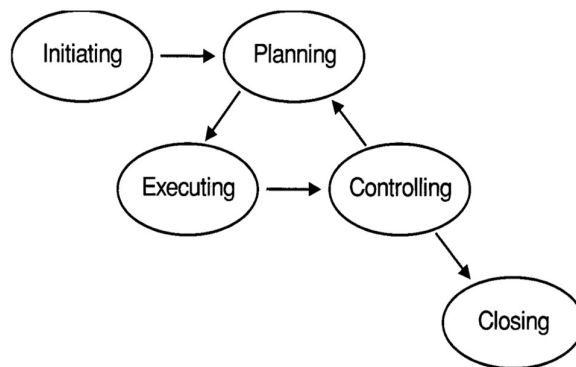
2.2 Project Management Life Cycle

Traditional project management outlines that every project goes through specific phases during its development, from start to finish, with each phase needing to be tracked and completed to ensure a successful outcome. These phases are easy to identify in traditional project management as tasks are executed sequentially, demanding substantial project planning in advance. What's more, each phase is considered final once executed, with no expectation of revisiting it (Hass, 2007). On account of this, the project life cycle serves as a foundational framework that project managers and their teams use to track these phases and systematically organize projects.

Although project life cycle phases can vary across different industries, the PMBOK® Guide from the Project Management Institute (2008) provides a standard definition explaining that *“project management is executed through the application and integration of the 42 systematically organised project management processes comprising the 5 Process Groups: initiating, planning, executing, monitoring and controlling, and closing”* (see Figure 2):

1. Initiation phase - sets the overall direction of the project. The focus is on defining the scope and objectives of the project, aligning stakeholders, and formalizing a project charter that defines primary goals and other essential details.
2. Planning phase - formulates a scheme for completing the project. This phase emphasizes precise scheduling, resource allocation, and risk mitigation.
3. Execution phase - focuses on implementing the project plan and achieving results.
4. Monitoring and Controlling phase - refers to monitoring the progress of the project and implementing corrective measures when necessary. This phase plays a vital role in ensuring that the project remains within the defined scope, budget and timeframe.
5. Closing phase - formalizes the completion and acceptance of the project through final approvals, evaluation and documentation.

Figure 2: Basic Project Management Processes



Source: Duncan (1993).

2.3 IT Project Management

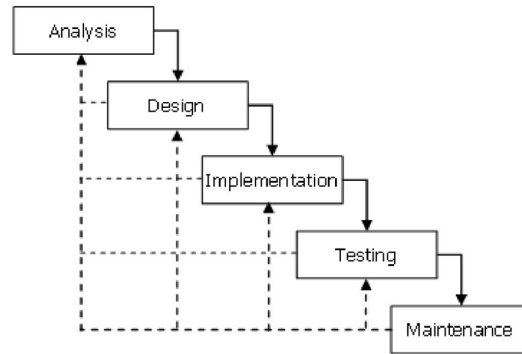
The IT industry is among the fastest-growing sectors globally, making it highly attractive to investors and creating opportunities for millions of new projects. While the core principles of project management apply across industries, IT projects are distinguished by their uniqueness and complexity, primarily due to the dynamic environment and rapid pace of technological advancement (Brewer, & Dittman, 2018). During the development, each IT Project goes through certain phases. They all belong to a process called System Development Life Cycle (SDLC) or Software Development Life Cycle, which actually enables the production of the software through initiation, planning, developing, testing and maintenance phase. These phases can have different names depending on the organization, they can also be combined into one, or split into multiple phases (Hughes et al., 2019). SDLC stands for a structured process used in systems and software engineering to create or modify information systems. Primarily associated with computer and information systems, SDLC serves as the foundation for various software development methodologies, which provide frameworks for planning and managing the development process (Kute, & Thorat, 2014). Over time, many variations of SDLC have been used for managing the project and this is the classification of the most used models:

2.3.1 Waterfall

It is believed that this approach came from Winston W. Royce, who first presented it in 1970 in his paper where he was explaining how to manage the development of large software systems (Royce, 1970). Waterfall represents a traditional SDLC model which follows a sequential process where each phase cascades into the next one, hence the name. Other authors also refer to Waterfall as the classic life cycle model, comprising fixed phases (see Figure 3): Requirement Analysis, Design, Implementation, Testing, and Maintenance (Gupta et al., 2021). These phases are distinct, non-overlapping and cannot be revisited once completed.

Deliverables, such as requirement specifications, design documents, and software architecture, serve as inputs to subsequent phases (Hughes et al., 2019). Additionally, in this model, the development is commonly planned using Gantt charts with the emphasis on stability, structured design, and adherence to predefined requirements (Arora, & Arora, 2016). This model is most effective when all requirements are clear, fixed, well-defined, and fully understood, with no ambiguity in the product definition or technology (Gupta et al., 2021). As the foundational model in software development, many other methodologies have evolved from it.

Figure 3: The Waterfall Model



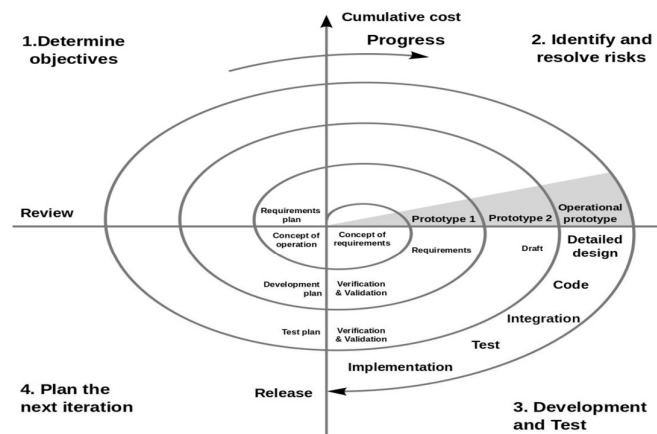
Source: Bassil (2011).

2.3.2 Spiral

The Spiral model represents another SDLC approach in development, firstly introduced by Barry W. Boehm during the late 1980's. This is an evolving risk-driven approach that provides a framework for guiding the software process, created by combining good features from different approaches and avoiding their flaws (Boehm, 1988). It offers great adaptability for a wide range of software projects and is particularly suitable for large, complex, and ambitious software systems, where addressing risks at each evolutionary stage is necessary (Mishra, & Dubey, 2013). The model includes phases represented on the diagram (see Figure 4):

1. Planning phase, where objectives are being determined
2. Risk analysis phase for risk identification and resolution
3. Product development and testing phase
4. Evaluation phase in which next iteration is planned

Figure 4: The Spiral Model

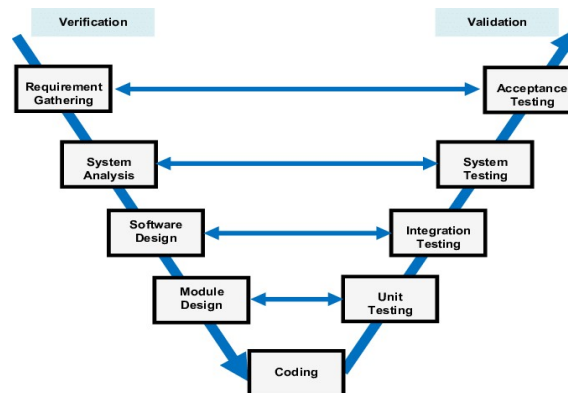


Source: Gupta et al. (2021).

2.3.3 V model

The V-shaped model, an evolution of the Waterfall model, adopts a structured and sequential development approach with a strong emphasis on rigorous testing at each stage (Gupta et al., 2021). Unlike the linear progression of the Waterfall model, the V-shaped framework aligns each development phase with a corresponding testing phase (see Figure 5), ensuring parallel processes of verification and validation (Arora, & Arora, 2016). This method is particularly well-suited for projects with clearly defined and stable requirements and when risk for changing is low. Primarily applied to small and medium-scale projects, the model's focus on early defect detection and correction through continuous testing increases the overall quality and reliability of the software (Alam et al., 2022). However, its effectiveness depends on the availability of an experienced technical team and appropriate resources, since this model is pretty rigid and inflexible.

Figure 5: V-Model in Software Development



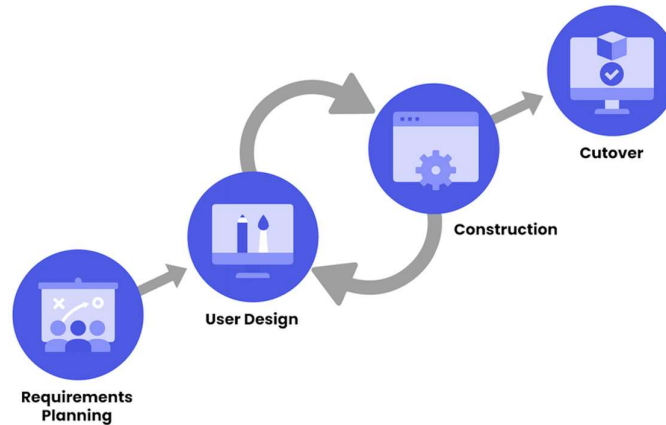
Source: Nakkasem (2020).

2.3.4 RAD

Rapid Application Development (RAD) model was first formalized in the early 1990s and it was designed to respond to the need to quickly develop and deliver systems in a short time period. This approach prioritizes iterative prototyping and continuous user feedback, rather than upfront planning. Typical development phases in this model are Requirements planning phase, User design phase, Construction phase and Cutover phase (see Figure 6), which includes implementation, data conversion, testing, transition to a new system and user training (Martin, 1991). Timeboxing is a fundamental principle in the RAD method, requiring completed tasks within a predefined and unchangeable timeframe. While requirements are prioritized and may be adjusted to fit the time constraints, the timebox remains strict. This approach is best suited for applications that emphasize user interface development and involve minimal complexity in requirements and computations (Mackay et al., 2000). Mishra & Dubey (2013) conclude that RAD requires a modularized system and

should be used on small or medium sized projects where developers and customers must remain dedicated to fast-paced activities within a reduced time frame.

Figure 6: Phases of Rapid Application Development



Source: Makadia (n.d.).

3 AGILE PROJECT MANAGEMENT

In today's fast-paced and rapidly evolving world, industries are experiencing constant change, which makes traditional approaches to software development such as the Waterfall method increasingly inadequate. To face new challenges, agile methods have emerged as a more flexible and adaptive alternative. Because it is designed to adapt to frequent changes in project requirements, agile enables teams to respond quickly to change, improve collaboration, and deliver high-quality software more efficiently. By embracing iterative development and continuous improvement, agile has become a cornerstone of modern project management, ensuring that development remains aligned with day-to-day changing business and user needs.

3.1 Defining Agile

The agile philosophy is grounded in the Agile Manifesto, a document that articulates four core values and twelve principles that define agile project management practices. The Manifesto was created in February 2001, as a result of a discussion between 17 experienced software development experts who formed the Agile Software Development Alliance. They all agreed that the primary goal is to ensure customer satisfaction by delivering high-quality products on time. This goal calls for a flexible, lightweight and team-oriented approach to software development, with a strong focus on culture and values.

The key values presented in the Agile Manifesto are prioritizing (Fowler, & Highsmith, 2001):

1. Individuals and interactions over processes and tools,
2. Working software over comprehensive documentation,
3. Customer collaboration over contract negotiation,
4. Responding to change over following a plan.

In addition, the Agile Manifesto is supported by twelve guiding principles, which provide a practical framework and a guide for implementing its four core values. These principles establish the foundation for agile methods, shaping decision-making, team collaboration, and iterative software delivery (Beck et al., 2001):

1. Customer Satisfaction - Prioritize timely and continuous delivery of valuable software.
2. Embrace Change - Adapt to changing requirements, even at late stages, to maximize customer value.
3. Frequent Delivery - Release working software regularly, preferably in shorter cycles.
4. Collaboration - Enabling business stakeholders and developers to work together on a daily basis.
5. Empowered Teams - Build projects around motivated individuals, providing support and trust.
6. Effective Communication – One-on-one conversation is the most effective way to share information.
7. Measure Progress through Software - Functional software is the key indicator of progress.
8. Sustainable Development - Maintain a consistent, long-term pace of development.
9. Technical Excellence - Focus on high-quality design and continuous improvement.
10. Simplicity - Maximize efficiency by minimizing unnecessary work.
11. Self-Organizing Teams - The best solutions emerge from independent and empowered teams.
12. Continuous Adjustments - Regularly review and refine processes for greater efficiency.

Constructed on these principles, various agile methods have emerged over time, such as Scrum, Kanban, Crystal, Extreme Programming (XP), and Feature-Driven Development (FDD), among others. The following chapters of this thesis will provide a comprehensive analysis of Scrum and Kanban, as they are the most widely utilized agile methods in practice, according to the 16th State of Agile Report (Digital, 2022).

Even though there are differences between the listed methods, they all have some common characteristics (Coram, & Bohner, 2005):

- Collaboration - To promote strong teamwork within the development team and with stakeholders,
- Code reviews - To improve code quality and knowledge sharing,
- Small teams - They enhance collaboration and minimize coordination overhead,

- Short release schedules - To deliver working software frequently,
- Time-boxing - Having fixed release schedules, with flexible feature scope to prevent scope creep,
- Constant testing - To ensure software quality all the way during the development.

Agile projects are typically organized into multiple iterative cycles, commonly known as Sprints, that allow for incremental development and continuous feedback from stakeholders. Each project is defined with a clear scope and is managed through a product backlog, which prioritizes tasks and features based on business value and project goals. The software development process within agile development usually follows a series of key phases, including requirements analysis, development, testing, and implementation. Hass (2007) argues that agile's core value lies in its flexibility and the emphasis on collaboration, with minimal upfront planning. She explains that by gradually building the system through increments and incorporating customer feedback at each stage, the agile approach enables continuous refinement of requirements, ensuring that the product better aligns with the organization's needs. Additionally, she claims that this approach minimizes modification costs by focusing on smaller, manageable units rather than major system changes.

What's more, agile embodies the principles of rapid responsiveness and adaptability, which are the defining characteristics of this SDLC model. Based on iterative and incremental development, agile focuses on collaboration and communication among self-organizing, cross-functional teams to gradually develop requirements and solutions over time, while ensuring customer satisfaction (Dora, & Dubey, 2013). Likewise, the agile method is convenient for feature-driven projects, with frequently changing requirements and close collaboration between team members and business users (Olorunshola & Ogwueleka, 2022). The model, along with its well-known frameworks, Scrum and Kanban, will be thoroughly analysed in the upcoming chapters.

3.1.1 Incremental development

This model could be compared to the Waterfall approach because of its phased structure; however, development and functionalities are delivered in distinct fragments, or increments (Hughes et al., 2019). The focus is on dividing the system into smaller, deliverable parts, completing them one at a time. Additional features are progressively added in subsequent stages, based on prioritized system requirements until the entire system is fully developed (Mishra, & Dubey, 2013). The incremental model provides flexibility to changes and makes a good fit for projects with high demand for quick product releases. This approach allows earlier stages of development to be revisited in order to incorporate stakeholder feedback, allowing for rapid resolution of defects and preventing them from accumulating until the final phase of the project.

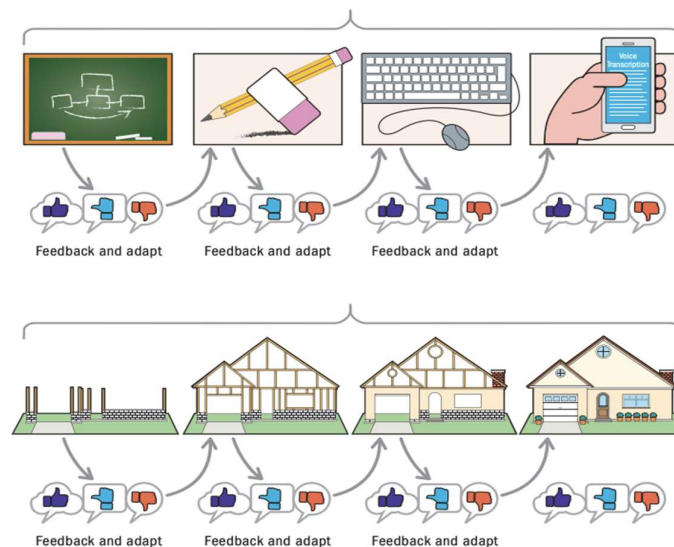
According to the PMBOK® Guide (Project Management Institute, 2021), incremental and iterative models are classified as hybrid methods that integrate both predictive and adaptive approaches (see Figure 7).

3.1.2 Iterative development

As the name suggests, this model delivers a product through a series of iterations or development cycles. The iterative model concentrates on progressively improving the system by repeating the phases of development multiple times until the final version meets the requirements. This approach is suitable for projects where initial requirements are not fully defined, allowing for modifications of functionalities during the development and the integration of new technologies into the ongoing project (Gupta et al., 2021).

A key advantage of this method is its ability to spot the risks early, similar to an incremental approach. By identifying and mitigating risks in initial iterations, corrective actions can be implemented in following iterations to minimize their impact. Iterative approach promotes flexibility, allowing teams to respond effectively to changing requirements while delivering functional software. Hughes et al. (2019) highlight the complementary nature of incremental and iterative models, explaining that software development can be divided into increments, and then each of them implemented through iterations.

Figure 7: Iterative and Incremental Development

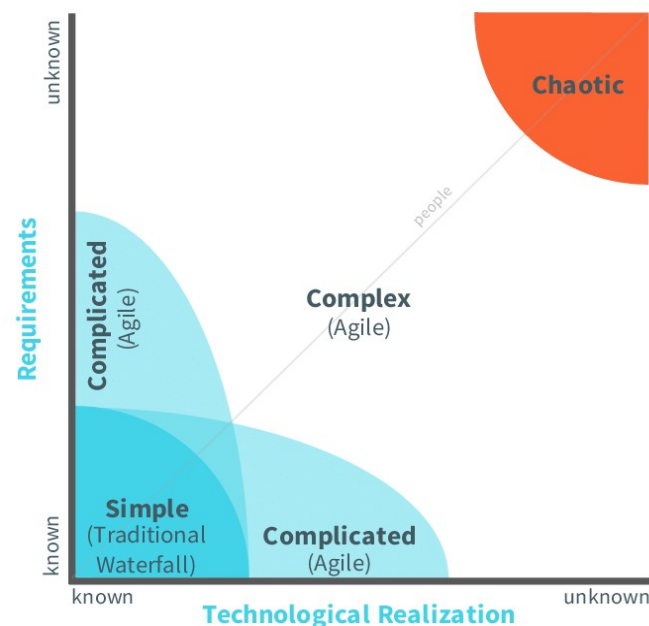


Source: PMBOK Guide (2021).

3.2 Comparative Analysis of Agile vs. Traditional Methodologies

In order to determine the most suitable method for a project, organizations rely on various frameworks to help making the decisions. One of the widely used tools is the Stacey Matrix, developed by Professor Ralph Douglas Stacey in the early 1990s. This framework helps organizations assess project complexity and uncertainty by evaluating two key factors: the level of agreement on requirements and the certainty of technology (Stacey & Mowles, 2016). Based on these factors, projects are categorized into four zones: simple, complicated, complex, and chaotic. Each zone requires a different approach of managing the work. Using the Stacey Matrix, helps teams to determine the most effective agile or traditional approach, ensuring better project outcomes.

Figure 8: Stacey Matrix adapted to Software Development



Source: Uludag et al. (2019).

As illustrated in the diagram (see Figure 8), the Traditional Waterfall approach is most suitable for projects with well-defined and stable requirements, where both the expected outcomes and the technological solutions are clear. This structured method is effective when there is certainty about both what needs to be built and how to build it. Meanwhile, at the opposite end of the spectrum, projects with unclear or constantly changing requirements, disagreement on how to solve the problem, and high levels of uncertainty may lead to chaotic situations, which should be avoided. Between these two extremes lie the complicated and complex areas. The complicated area represents a predictable but less straightforward environment, where requirements may be somewhat unclear but still manageable, and the technology involved is familiar. In some cases, projects may have clear requirements but involve unfamiliar technologies, which requires additional investigation. In such cases, agile

methodologies are recommended due to their iterative approach and continuous feedback mechanisms, which allow teams to adapt as they learn more. The complex area, which occupies the largest area in the diagram, is characterized by high uncertainty and risk. Software development serves as a typical example of a complex environment where requirements change as the project progresses, and optimal technical solutions may not be that obvious immediately. In such scenarios, traditional methodologies often fail due to their rigid structure. Instead, teams need an exploratory approach that prioritizes transparency, continuous inspection, and iterative adaptation. Agile frameworks such as Scrum are particularly well-suited for these environments, because they enable flexibility, responsiveness, and incremental progress despite the uncertainty.

3.3 Advantages of Agile

The benefits of agile methods have been widely recognized in the literature, with researchers highlighting their obvious benefits. As mentioned earlier, agile came out as a response to the limitations of traditional methods, which were better suited to large-scale, stable projects with clearly defined structures and fixed requirements. In contrast, agile approaches are particularly effective in dynamic and complex environments, where adaptability and iterative development are crucial (Highsmith, 2009).

Coram and Bohner (2005) state that agile methods effectively manage the uncertainty and complexity of modern software development through proven principles that reduce risk and enhance productivity. When applied individually, they improve communication and efficiency, and what's more, when combined with other agile methods, they create synergies that further optimize project results.

Lutwama et al. (2024) conclude in their paper that agile methods enhance flexibility and accelerate development by enabling teams to respond quickly to customer feedback and evolving requirements. Their adaptability allows teams to choose sub-methods that best fit their needs — such as Scrum, with its structured roles and events, or Kanban, which supports continuous delivery and workflow efficiency.

3.4 Disadvantages of Agile

Although agile methods have proven to be highly effective across various projects and industries, they are not universally applicable. In some situations, agile may not be the most suitable approach, as it comes with its own set of limitations and challenges.

As believed by Sharma et al. (2012), agile methods, while promoting flexibility and customer collaboration, carry several challenges. Their reliance on clear information from customers can lead to project misalignment if requirements are unclear. Minimal documentation, although effective in saving time, complicates project implementation and

long-term maintenance due to frequent design changes. Constantly changing requirements can also lead to wasted time and effort if previous iterations are discarded. Moreover, although the agile method supports managerial decision-making, it can burden developers with continuous adaptation under tight deadlines.

Similar to previous authors, Pargaonkar (2023) also concludes that while agile promotes flexibility and close collaboration, it can be demanding in terms of resources due to the need for continuous engagement from both team members and stakeholders. Using and managing agile methods in large or complex projects can become challenging, as frequent iterations, testing, and changes can complicate coordination and delay delivery. Moreover, the reduced emphasis on detailed documentation, compared to traditional methods like Waterfall, can lead to incomplete records of key decisions in design. Agile methods also rely heavily on continuous input from customers, and any lack of availability or clarity from customers can hinder progress. Furthermore, the fluid nature of requirements increases the risk of scope creep, potentially overusing limited resources and impacting quality. Finally, agile's strong focus on speed and quick delivery can sometimes lead to compromises thorough testing and overall software quality.

4 SCRUM

The term scrum was originally introduced by Hirotaka Takeuchi and Ikujiro Nonaka. In their paper titled “The New Product Development Game” published in the Harvard Business Review, 1986, they describe the transformative changes that were happening in the rapidly changing market. They highlighted the burgeoning significance of speed and flexibility as important factors for companies who wanted to gain competitive advantage, apart from the traditional principles including high quality, low costs and differentiation. As a result, a different approach was needed and that’s how the duo came up with the initial concept of Scrum. It is important to say that they didn’t invent this term, it was already present in rugby representing teamwork between the players. Inspired by this practice, they used it to describe the product development process: Under the rugby approach, the product development process emerges from the constant interaction of a hand-picked, interdisciplinary team whose members work together all through, from start to finish. As an alternative to moving in defined, highly structured stages, the process is born out of the team members’ cooperation (Takeuchi, & Nonaka, 1986). Understanding this, Scrum emerges as the dynamic framework empowering teams to consistently refine and enhance their product via iterative cycles. As its adoption spread, diverse interpretations of Scrum surfaced among different authors and the reason for that can be attributed to the fact that Scrum represents a simple, lightweight and highly flexible approach that is constantly evolving and is suitable to be used in various projects across diverse industries.

Scrum was first introduced and formalized as a framework for software development at the OOPSLA conference in Texas, in 1995 by Ken Schwaber and Jeff Sutherland. Later, they

created Scrum Guide where Scrum was well-defined as a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value (Schwaber, & Sutherland, 2011). In the later edition, they reformulated it as a lightweight framework that helps people, teams and organizations create value through tailored solutions for complex problems (Schwaber, & Sutherland, 2020). Another definition describes Scrum as an agile approach for developing innovative products and services (Rubin, 2012). Conboy (2009) states that Scrum emphasizes creating business value through frequent customer interaction and accepting business dynamics. Furthermore, Scrum as an agile framework provides just enough structure for people and teams to integrate into how they work, while adding the right practices to optimize for their specific needs (What is Scrum, n.d.). According to Abrahamsson et al. (2002) Scrum concentrates on how the team members should function in order to build the system flexibly in a highly dynamic environment. In the words of Cervone (2011), Scrum deliberately uses iterative and incremental processes within a collaborative team structure, primarily to address the challenges of fast-paced agile environments, manage conflicts, enhance communication, foster cooperation, and shield against disruptions, ultimately striving to deliver a more tailored product compared to traditional approaches. To summarize, Scrum is designed to handle frequently changing project requirements by improving communication and encouraging close collaboration between the project owners, developers and other team members.

4.1 Defining Scrum roles

Scrum framework dictates three specific roles or accountabilities needed in a Scrum Team for a successful realization of the project (Schwaber, & Sutherland, 2020). Those roles are Product Owner, Scrum Master and Development Team. The product owner defines the priorities and order of development; the ScrumMaster guides the team in creating and sticking to their process within the Scrum framework and the development team is responsible for choosing how to deliver the features requested by the product owner (Rubin, 2012). Practice in this field states that the Scrum Team should ideally consist of ten or less individuals, in order to be more efficient and optimize productivity. It's essential to understand that Scrum teams embrace a collective accountability model, where success or failure within a project isn't attributed to one role alone (Scrum Institute, n.d.).

4.1.1 Product Owner

As the name itself says, the Product Owner is someone who primarily owns the product by defining the product vision and roadmap. Product Owner has a crucial role in translating stakeholder's requests and needs into actionable directives for product implementation. In other words, they serve as the intermediary between the customer, who determines value, and the development teams responsible for implementing that value into functional software (Project Management Institute, n.d.). Moreover, while they may not write the detailed tests

themselves, they make sure that high-level tests are in place to confirm when a feature is truly complete. Thus, the product owner serves as a blend of a product manager, business analyst, and tester, facilitating effective communication and ensuring project success (Rubin, 2012). Additionally, as Scrum Guide states, the Product Owner is responsible for maximizing the product's value delivered through the work and effort of the whole Scrum Team. To conclude, this role includes numerous responsibilities, some of previously mentioned:

- Determining and clearly communicating Product Goal and Vision
- Creating and prioritizing Product Backlog items
- Ensuring transparency and understanding of requirements
- Utilizing decision-making skills to navigate the product towards its goal
- Collaborating with both Scrum Team and key business stakeholders

4.1.2 Scrum Master

The primary function of scrum master is to ensure a proper understanding and implementation of Scrum method by all team members, including the stakeholders outside the Scrum Team. Positioned as the one accountable for team effectiveness, the Scrum Master holds the responsibility to empower team members to regularly refine and enhance their practices within the Scrum framework (Schwaber, & Sutherland, 2020). It is of a great essence for the Scrum Team to designate someone in charge of removing blockers and other distractions, coordinating the activities and helping with overcoming obstacles, so the team can concentrate on delivering value and meeting the Sprint goal. Moreover, Scrum Master serves as a link between the organization and Scrum Team, and as a facilitator during Scrum Events like Daily Scrum, Sprint Planning, Sprint Review and Sprint Retrospective.

Correspondingly, Rubin (2012) lists the principal responsibilities of a Scrum Master as:

- Coach - guiding the team in understanding and effectively applying Scrum principles and practices
- Servant leader - supporting the team by removing obstacles and barriers, facilitating collaboration, and making sure they are equipped with resources and support to achieve their goals
- Process authority - by having knowledge about Scrum practices, Scrum Master is ensuring that the team adheres to them while also continuously improving the processes
- Interference shield - acting as protection against outside interruptions or distractions, helping the team stay focused on their work
- Impediment remover - identifying and resolving any obstacles that slow down the team, so they can work effectively and accomplish their goals
- Change agent - encouraging a culture of continuous improvement and helping the team adapt to frequent changes in the environment or requirements

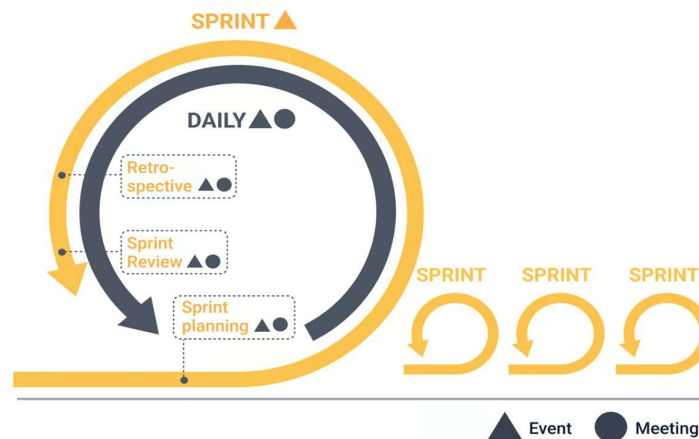
4.1.3 Development Team

Last role, or better said the set of roles, the final cornerstone in the Scrum Team makes the Development Team. This team consists of individuals equipped with diverse skill sets essential for delivering the product and materializing the Product Owner's vision. Some important characteristics of this team are that it is cross-functional, self-organizing, flexible and treats each member as equal. Alongside developers, the team is composed of designers, QA Engineers, Business Analysts and others, holding a different level of experience, different background and expertise (Rubin, 2012). The Development Team is responsible for creating the product and delivering value through Sprints until the final version is complete. Their collaboration and dedication drive steady progress toward the project's objectives, highlighting both the effectiveness and adaptability of the Scrum framework.

4.2 Defining Scrum activities

Scrum comprises specific activities, often referred to as events by various authors. The Scrum framework delineates five fundamental events defined in Scrum Guide, each fulfilling a distinct purpose: The Sprint, Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective (Schwaber, & Sutherland, 2017). It's important to note that these activities, or events, are time-boxed and take place regularly at consistent time and location to simplify complexity and team coordination (see Figure 10).

Figure 9: Scrum Events and Meetings



Source: Wranek (2023).

4.2.1 The Sprint

In Scrum, the whole project is divided into short, fixed-length, repeating iterations, called Sprints. These iterations usually last one to four weeks, and are pretty consistent. The start and the end of the Sprint are always set and there should be no changes time wise, nor scope wise, but in highly dynamic environments and businesses, sometimes this is inevitable. Each

Sprint should have a Sprint goal, which will bring some value and get the team closer to the Product goal. Rubin (2012) too agrees that sprints should produce a potentially shippable product increment that is completed in conformance with an agreed-upon definition of done. Sprint includes other activities that are happening in these two weeks (or a month), such as Sprint Planning, Daily Scrum, Sprint Review and Sprint Retrospective. Scrum team may also use reports and charts like burndown, burnup, velocity report and cumulative flows, during and after the Sprint, to get insights and track the progress, which should enable prompt and punctual project delivery, if used wisely. Practice showed that shorter Sprints are simpler, require no big effort to plan and are bringing more learning experience and easy improvements by providing fast feedback and quick detection of failures.

4.2.2 Sprint Planning

The first step for starting a new Sprint is called Sprint Planning. This event gathers the whole Scrum Team with the purpose of defining the scope of the sprint, setting the Sprint goal and the best way to achieve it. Before the Sprint Planning session starts, Product Backlog items should be properly prepared and prioritized, making it easier for the team to select which items are going to be included in the Sprint Backlog. The development team determines the specific product backlog items aligned with the previously set Sprint goal and agrees that these items can be realistically delivered within the timeframe of the Sprint (Rubin, 2012). The second phase in Sprint Planning consists of estimating the work and the effort needed for the selected items and breaking down these items into smaller and simpler tasks, like user stories. Each user story contains a brief description that is used for planning and facilitating discussions, alongside acceptance tests that establish the criteria for defining when the story is done (Mahnič, 2015). Traditionally, the effort was given in time format: hours, days, weeks, needed for a task to be completed. Nowadays, story points are being used as units to show the required effort (Radigan, n.d.). Hence, the story points are assigned to each task relatively, based on the task scope, complexity, uncertainty, etc.

Scrum teams can use various techniques for estimations such as:

- Planning Poker (by using the Fibonacci Sequence) - employing a deck of cards featuring Fibonacci numbers, team members individually estimate the effort required for user stories, followed by a collaborative discussion to reach a consensus estimate
- T-shirt Size Estimation (XS, S, M, L, XL) - by using T-shirt sizes as a relative indicator, user stories are sized to quickly estimate their effort or complexity without getting into numerical details
- Bucket System - user stories are sorted into predefined buckets according to their estimated size or complexity. This approach simplifies estimation and helps with prioritizing work.
- Voting - team members individually estimate the effort or complexity of user stories, then discuss differences to reach a consensus

- Discussion - team members openly communicate and share perspectives on the complexity of user stories, leveraging their expertise to reach a shared understanding and consensus on estimates

4.2.3 Daily Scrum

Every day through the working week, Scrum Team is gathering in a meeting called Daily Scrum meeting, or a stand-up. These meetings are usually short and held at the same time and place, facilitated by the Scrum Master, with the purpose of organizing the work, tracking the Sprint progress, identifying potential blockers and enabling better communication within the team. The meetings are designed so that each team member is giving an update on what has been done the previous day, what is planned to be done next and additionally pointing out if they came across any issue (Scrum.org, n.d.). Explained by Rubin (2012) Daily Scrum presents an inspection, synchronization, and adaptive daily planning activity that helps a self-organizing team do its job better. Cervone (2011) states that the purpose of the daily Scrum is to both keep an eye on team's progress as well as allow team members to take responsibilities for their work and commit to each other and the Scrum master so that work can proceed without blockers, in the most expedient manner.

4.2.4 Sprint Review

Sprint Review meeting comes at the end of each Sprint and it represents one of the five activities in the Scrum framework, primarily focused on the product. The goal for this meeting is to evaluate the previous work, showcase the achievements and gather feedback (Scrum Alliance, n.d.). The Scrum Team is in charge of presenting the features that are being completed in the Sprint to other stakeholders in order to get more information about the product's progress and to determine if the Sprint has met their expectations and contributed to long-term goals. The outstanding features are being re-estimated and altered for the future Sprints. Finally, by conducting Sprint Reviews properly, it is ensured that the product development is going in the right direction.

4.2.5 Sprint Retrospective

Sprint Retrospective usually represents the last activity that takes place in the Scrum Framework sequence and concludes the Sprint. The main difference between Sprint Review and Sprint Retrospective meeting is that the latter one puts focus on individuals and the team, the interactions between team members, the processes and tools being used, but not so much on a product itself. The goal of this meeting is to help Scrum Team identify and discuss what went well during the Sprint, what kind of issues they faced, but also to suggest some improvements and action steps which will lead the team to perform better. To summarize, Sprint Retrospectives are there to enable collaborative learning, act as catalysts for change, and generate action (Derby, & Larsen, 2006).

4.3 Defining Scrum artifacts

The artifacts within the Scrum represent the source of work, including all key information and details needed for a successful product development (Schwaber, & Sutherland, 2020). There are three artifacts defined by Scrum Guide named Product Backlog, Sprint Backlog and Product Increment. As indicated by Fowler (2019), the purpose of these Scrum artifacts is to facilitate, support, and fulfil the requirement for transparency. In other words, they are envisioned to serve as a guide for the team from the big-picture plan in the Product Backlog to the specific tasks in the Sprint Backlog, leading to visible progress in the Product Increment.

4.3.1 Product Backlog

One of the Scrum principles is to always prioritize tasks and primarily work on the most critical things. In order to do that, the Scrum Team needs a Product Backlog. Product Backlog stands for a prioritized list of features required for achieving a goal, previously set by Product Owner and the Scrum Team. It can also be explained as a single source of requirements, containing features aligned with a Product Owner's vision (Rubin, 2012). It is crucial to have some form of guidance in place, an idea, or as previously emphasized - a vision, which needs to be converted into the product. This happens through the Product Backlog, composed of features that are translated into user stories. User stories can be described as core components in an agile framework. They are shaped as the smallest unit of work with the goal of delivering value to the product, from the user's perspective (Rehkopf, n.d.). A simple and effective way to describe user stories through three critical aspects: Card, Conversation, and Confirmation (3Cs) is offered by Jeffries (2001). User stories are represented on cards with minimal text to identify requirements, used for planning and supplemented with notes reflecting priority and cost. Conversations between customers and programmers drive understanding and clarification of requirements, supplemented with examples, particularly executable ones, which they call confirmation. Confirmation, provided by the acceptance tests, ensures clarity and certainty in implementation, allowing teams to deliver what is needed. A set of related user stories is called an Epic, which represents a high-level structure of set objectives and is used to better organize the Product Backlog into meaningful sections.

Later, as the product journey progresses, the Backlog transforms into a dynamic repository containing not just the initial features, but also additional elements like new features, refined design improvements, changes to existing features due to optimization and bugs (representing unexpected errors or problems in the product that need fixing). Product Backlog is not static, it is endlessly evolving, following the product development, but what is constant is the rule that the most important user stories or tasks are always prioritized on top of the Backlog and the ones with the lowest importance are at the bottom. In pursuit of keeping the Backlog healthy and ready for the upcoming Sprint(s), it is important to regularly

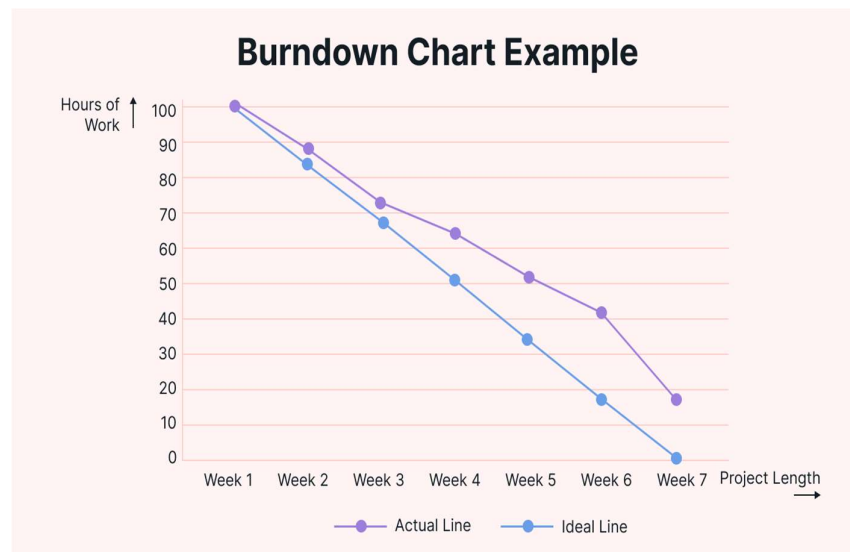
review and readjust the backlog items. This process is usually done in Product Grooming sessions, also known as Backlog Refinement, where Scrum team gathers and makes sure that Backlog items are properly assessed and realigned.

4.3.2 Sprint Backlog

According to Rubin (2012), the Sprint Backlog serves as a roadmap for the Scrum Team, composed of a list of detailed tasks required for designing, building, integrating and testing a group of features selected from the Product Backlog for that specific Sprint. Although it is recommended for the Sprint scope not to change frequently, Schwaber and Sutherland (2020) believe that the Sprint Backlog should be adaptable in order to keep up with all changes and disruptions that may occur during the Sprint, as long as it remains aligned with the Sprint Goal defined in the Sprint Planning session. Sprint goal serves as an objective and the business value that the product increment should deliver (Asproni, 2006).

Sprint Backlog items are estimated and allocated to each member of the team depending on the team's capacity. During the Sprint, the team updates the Sprint Backlog by recalculating the estimated work remaining for each item, generating the data for a Burndown Chart (see Figure 11), a graph illustrating the remaining workload to be accomplished by the end of the Sprint.

Figure 10: Burndown Chart Example



Source: Motion (2023).

4.3.3 Product Increment

After each Sprint has been executed, the project team must produce a working and potentially releasable version of the new software product (Cooper, & Sommer, 2016). Thus,

a product increment represents an output, where each item meets the team's definition of done, otherwise the item cannot be released and should be brought back to Product Backlog. Definition of done can be simply described as a checklist stating which requirements need to be satisfied in order for Product Backlog items to be translated as a valuable Product Increment. Definition of Done is essential because it assures a balance between short-term delivery of features and long-term product quality (Silva et al., 2018).

4.4 Advantages and disadvantages of Scrum framework

The Scrum method is classified as a first-generation agile method and as affirmed by the 16th State of Agile Report (Digital, 2022), it continues to reign as the foremost choice and the most widely embraced framework across industries. Its lasting popularity and widespread adoption highlight how effective it is in supporting iterative and collaborative approaches to product development. Moreover, Scrum method is best known for its simple and easy approach to software development (Schwaber & Beedle, 2001). Central to its philosophy is the emphasis on generating business value through regular engagement with stakeholders, as underscored by Conboy (2009). This commitment to adaptability and rapid response to changing business needs sets Scrum apart as a dynamic and customer-centric framework. Furthermore, Scrum ensures efficiency and effective use of time and money and also allows changing product requirements during the development. That is to say, Scrum's advantages are best manifested in dynamic environments where speed, rapidly changing technology, evolving requirements and competitive demands require flexibility and instant product evolution (Vlaanderen et al., 2011). Another important thing is the integration of the testing process within the Scrum framework which brings several benefits, including early detection of issues, improved collaboration, and better product quality. By synchronizing testing with development sprints, this approach promotes quick feedback, thereby minimizing the occurrence of bugs and ensuring better reliability and overall quality of the software product.

Considering that no method is without its imperfections, it's important to recognize that the Scrum framework, although very effective, is not without certain limitations. One significant challenge is its dependency on team members with high experience. The fundamental principle of a Scrum Team revolves around autonomy and self-governance, which can be intimidating for individuals who are relatively inexperienced in the method. Therefore, the integration of less experienced members into a Scrum Team may require additional support and guidance to ensure they are successfully adapted and can contribute to the team's goals. What's more, teams can sometimes be more focused on completing the sprint than delivering a high-quality product and this can result in prioritizing speed over quality. Another important thing that can represent a barrier for Scrum framework is the rigidity and resistance to change in the organizations (Atrey, 2023) and on top of that, Scrum is not the best fit for bigger teams, requiring too much involvement and coordination. Ganev (2018) states that

poorly defined tasks may result in inaccuracies, potentially causing sprints to exceed their originally estimated duration.

5 KANBAN

Taiichi Ohno, an engineer in Toyota and later an executive vice-president, introduced Toyota's lean production system in the late 1940s, based on two pillars: Just-In-Time production and Automation with a human touch, also called *autonomation* (Ohno, 1988). Kanban represented a production control system, developed to support Just-In-Time production and making full use of workers' capabilities (Sugimori et al., 1977). The Kanban system, alternatively known as the "Supermarket system" derives its conceptual basis from Taiichi Ohno's observation and adaptation of operational practices found within the American grocery retail sector (Ohno, 1988). The word "kanban" comes from the Japanese language and it translates as a visual card, or a signboard. Each card represents a specific task, and the total number of cards reflects the system's capacity. Work can begin only when a card is available, and as tasks move through the process, a card is assigned to each one to track its progress through the system. New work is queued until a card becomes available, ensuring a balanced workload.

In Toyota, they managed to achieve a Just-In-Time system and prevent overproduction by using these visual cards (displays) transmitting different types of information, from inventory levels to production volumes. By way of explanation, Kanban is a prompt-response scheduling system which passes on information between processes, with the goal of preserving production at an optimal level and promoting continuous improvement (Toyota Blog, 2013). Put simply, Kanban serves as the cornerstone of the *kaizen* (continuous improvement) process employed at Toyota, functioning as the pivotal mechanism driving its efficacy (Anderson, 2010). Since its successful implementation in Toyota, Kanban has been used in various industries, including healthcare, clothing industry, aeronautics, human resources and software development (Ahmad et al., 2018).

5.1 Kanban framework overview

The beginnings of Kanban's first implementation and integration into software development is attributed to David J. Anderson (Anderson, 2010). Anderson was asked in 2004 by Microsoft's IT department, to support one of their small teams, known as XIT Sustained Engineering, in enhancing workflow visualization and constraining workflow capacity (Kirovska & Koceski, 2015). During development, he recognized that the pull system they were implementing functioned as a virtual Kanban system, which led to the use of the Kanban method. In the subsequent years, Anderson and his colleagues dedicated efforts to refine their virtual Kanban implementation, incorporating additional functionalities.

In addition to the 3 core Kanban properties (Visualization, WIP limits, and Manage flow), they defined 3 additional properties: Make process policies explicit, Implement fast feedback loops, and Drive improvements based on methods and models.

The following list briefly explains each of these properties:

- Visualizing workflow - involves using a Kanban board to organize and display tasks represented on cards, as they move through various stages of the workflow, shown as columns on the board. Visualizing the workflow is important because it makes it easier to monitor progress, identify bottlenecks, and manage work more effectively.
- Limiting work in progress - requires setting clear limits on the number of tasks for the team, which minimizes multitasking. This ensures streamlined operations, and encourages team members to complete their current tasks before taking the new ones, which leads to increasing productivity and efficiency.
- Managing flow - this property focuses on clearly defining the scope of work and tracking its progress, with the purpose of optimizing the workflow and ensuring that resources are used effectively.
- Making process policies explicit - includes clear explanation and communication of work policies in the team, such as criteria for task movement, prioritization, and definitions of "done," which are essential for transparency and consistency, and moreover, making sure all team members understand and adhere to the same standards.
- Implementing fast feedback loops - this property is highly significant as it fosters evolutionary change, ongoing improvement and organizational alignment.
- Drive improvements based on methods and models - relates to employment of methods and models needed for optimizing processes and bringing continuous improvement to the organization

Following the Agile 2007 conference held in Washington, D.C., Kanban method began to surface and gained traction within the wider community (Anderson, 2010). Eventually, these 6 core properties evolved and expanded into the 6 General practices of Kanban. Gradually over the time, other authors provided their own definitions of the method. Vijay Anand & Dinakaran (2016) explain that the Kanban method uses an in-development limited pull system as the core mechanism to uncover system operation (or process) problems and stimulate collaboration to continuously improve the system.

Furthermore, Ikonen et al. (2011) state that Kanban method serves as a mechanism for the visual representation of task progress, attempting to unravel and clarify potential inefficiencies within the workflow. It also emphasizes rapid process adaptation through shorter feedback loops, prioritizing flow over mandatory iterations. To conclude, the Kanban processes effectively prioritize critical tasks requiring immediate attention, while mitigating the risk of incomplete work and enhancing flexibility across project tasks (Lei et al., 2017).

5.2 Kanban principles

Anderson (2010) stated in his book that the main reason for adopting Kanban is change management. Based on that, Kanban Change Management principles appeared as:

- Start with what you do now - the Kanban method encourages improving existing processes gradually instead of restructuring them completely, highlighting the value in preserving what's already working well within an organization
- Agree to pursue improvement through evolutionary change - the organization or team should recognize the need for a moderate, evolutionary method of improvement, especially if past attempts at significant changes have faced resistance, or if organizational dynamics discourage such initiatives. Without consensus on adopting a slow, incremental approach, there may not be sufficient support or friendly environment for implementing a Kanban initiative
- Stimulate leadership roles at all levels - to achieve progress, effective leadership must be the catalyst in initiatives for change. It is important to empower individuals while creating a culture of safety in order to enable rapid, data-backed changes without fear of repercussions

Equally important, the Kanban method stands up for adopting a service-oriented perspective to comprehend organizational dynamics and workflow. This paradigm views the organization as a dynamic network of interrelated services, guiding improvements in service delivery based on underlying principles, which might be initially overlooked by organizations who are lacking a customer-centric culture. Hence, Anderson defined the following set of Kanban service-oriented principles:

- Understand and focus on customer needs and expectations - it is important to acknowledge the customer's central role in the process, while focusing on meeting their needs and ensuring satisfaction. Kanban is about proactive understanding and communication with customers to ensure that their requirements are met and well pleased
- Manage the work; let people self-organize around it - the emphasis on process over people creates a more human-centric approach by relieving individuals from constant scrutiny and micromanagement, promoting self-organization and acts of leadership at all levels
- Evolve policies to improve customer and business outcomes - recognizing the interdependence with other stakeholders, collaboration is crucial while working within a network of correlated services. Through regular feedback, it is essential to ensure alignment between company's set of policies and the customers

5.3 Visualize workflow

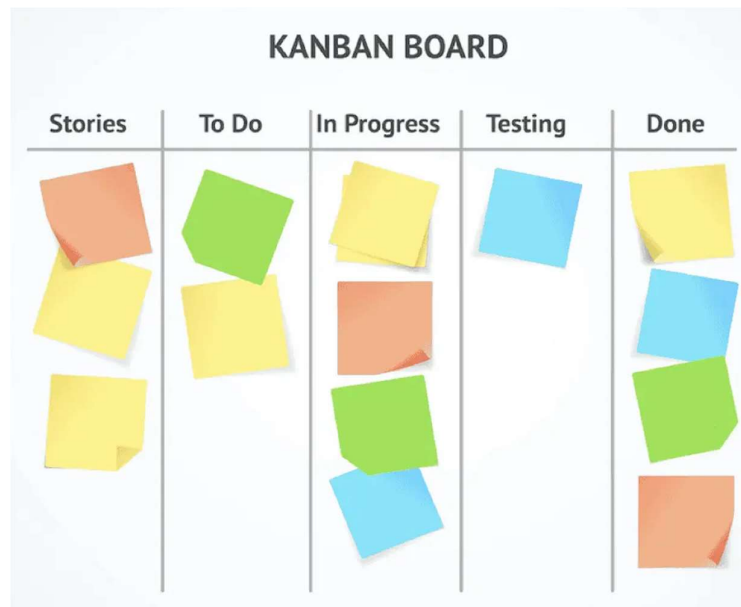
The concept of a "virtual Kanban board" was introduced to enhance visibility of Work-In-Progress (WIP) and identify constraints, while also placing a limit of one item in progress at

a time. Unlike traditional software development methods where tasks are "pushed" through the development pipeline, in this approach, individuals are assigned only one task at a time, and work is "pulled" into progress once the previous task is completed (Kirovska, & Koceski, 2015).

The Kanban board presents a visual tool for teams to help them organize project tasks, workflows, and communication. It facilitates efficient task management by providing a clear overview of each step in the project process (Hennigan, & Bottorff, 2024). This is done through split columns that are illustrating the workflow (new tasks, tasks in progress and completed ones) and sticky notes or physical cards representing the work items such as user stories, features, epics, bugs, improvement suggestions or other requirements. Kanban cards and boards are flexible and could be digital or physical. The colour of the card may indicate certain attributes or the current status. The content displayed on the card may vary and can consist of the project's initiation date, deadline (if applicable), a brief description of the task, its priority level, the assigned developer, and additional relevant details (Corona, & Pani, 2013). For example, user story can briefly describe a work item on a card or sticky note, stating what needs to be done, who it's for, and why it's being addressed.

A quick glance at the Kanban board (see Figure 12) reveals the project's overall status, while the dynamic movement of task cards reflects the ongoing progress or any potential bottlenecks as time passes (Ikonen et al., 2011).

Figure 11: The Kanban Board



Source: Cathy (2020).

5.4 WIP limits

One of the fundamental Kanban practices for ensuring the effectiveness of the pull-based system is to limit Work-In-Progress (WIP). Implementing WIP limits makes it possible to facilitate a steady flow of development, foster continuous progress and optimize resource utilization with the intention of minimizing the waste. Restricting WIP is resulting in elevated product quality and enhanced team efficacy. The combination of streamlined workflow and higher software quality contributes to shortened lead times, opening the door for frequent releases that enrich customer trust (Ahmad et al., 2013). In practice, this implies that each team member should finish the task they are currently working on before pulling another work item in the next stage. Without clear limits on Work-In-Progress and mechanisms to signal the introduction of new tasks, the system cannot be considered as adhering to the Kanban approach (Anderson, 2010).

5.4.1 Little's law

The relationship between WIP and lead time has been explained by a mathematical law called Little's law. Little's law is a principle widely acclaimed in manufacturing, which establishes a linear relationship between the quantity of work in progress and the average lead time.

Three key measures of process performance in Little's law are:

- WIP (Inventory) - the total number of items present within the process boundaries
- Throughput (Flow rate) - the number of items moving through a specific point in the process per unit of time
- Cycle time (T) - the total time spent by an item within the process boundaries

$$I = R \times T \quad (2)$$

Essentially, this equation (2) shows that increasing the number of work items in the process prolongs the time required to complete each item. Therefore, setting appropriate WIP limits at each stage of the process is crucial for achieving shorter lead times. Teams typically set WIP limits at the project's beginning, adjusting them as needed throughout the process to enhance efficiency and prevent excessive waiting (Damij, & Damij, 2021). To clarify, an inadequately estimated WIP limit can lead to suboptimal outcomes. A limit set too high may foster idleness within the workflow, as tasks stagnate without due attention, lacking designated owners. On the other hand, setting the limit too low may lead to idleness among team members, as all available tasks are concurrently in progress, leaving some individuals without assignments.

The WIP limits are fulfilling two main purposes in optimizing the workload. Firstly, they confine the amount of work subject to changing priorities or requirements, enabling

immediate responses and minimizing waste. Secondly, WIP limits control the workflow to match the pace of the slowest step, maximizing efficiency and productivity. It's advisable to start with the modest WIP limits for each step, ensuring full team engagement while preventing work backlogs. These limits can be adjusted iteratively to optimize throughput.

5.4.2 Different approaches in setting the WIP limit

Brechner's (2015) approach involves determining WIP limits by assessing the throughput of the slowest step within the workflow and subsequently adjusting the throughput of other steps to match this pace. To optimize team engagement in delivering value to customers, Brechner recommends implementing the minimal WIP limits necessary. First step dictates establishing the WIP limit for the slowest step and incorporating a 50 percent buffer. This phase ensures the slowest step remains constantly active, while effectively restricting the number of work items at that stage. In the following phase it is important to use proportional ratios to determine WIP limits for the remaining steps, so that their throughput is aligned with the slowest step. The results of this approach provide initial values that can be continually refined to maximize overall throughput. This approach is explained more thoroughly in the following example:

- On average, each designer can create roughly eight designs per month, each developer can develop roughly four items per month, and each quality assurance (QA) specialist can test roughly five items per month
- Developing was identified as the slowest step, with a throughput of four items per month per developer
- The team consisted of four developers working on items from the backlog
- The throughput was sixteen items per month (4 developers x 4 items per developer)
- Dividing that throughput by the average rates for each step indicates the number of people needed for each step to match sixteen items per month (two designers, four developers, and three QA specialists)
- By adding a 50 percent buffer to these totals and rounding up, the WIP limits for each step are determined as: 3 for Design, 6 for Develop, and 5 for Test

Additionally, there is the floor-ceiling approach which utilizes two different strategies: the floor strategy, which begins with conservative limit values and gradually increases them as necessary, and the ceiling strategy, which starts with higher limit values and reduces them as needed (Colucci, 2016). With that being said, the conclusion is that regardless of the chosen approach, refining and implementation of WIP limits is a dynamic process that requires ongoing assessment and adjustment.

5.5 Manage flow

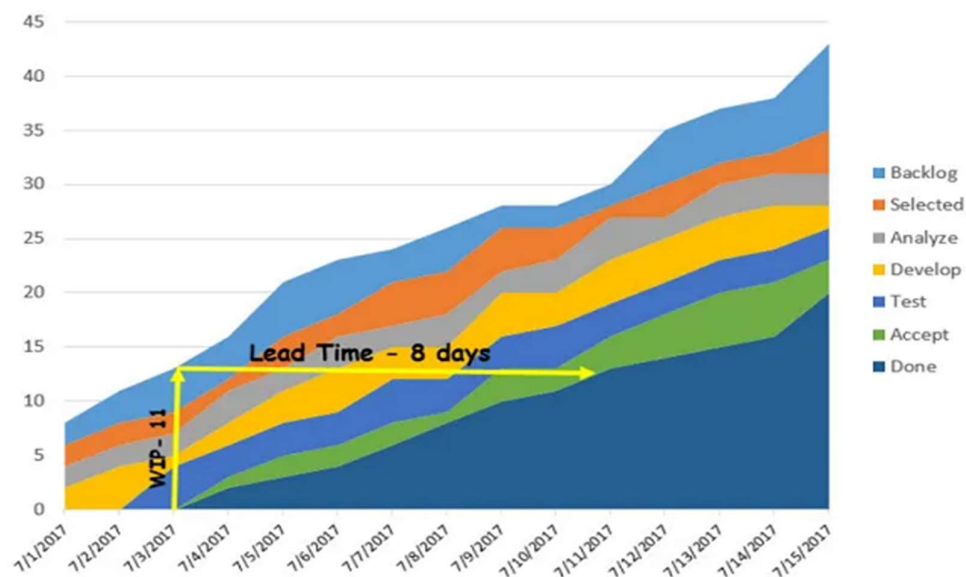
For decades, Toyota's production philosophy has been based on the principle of continuous flow. This approach ensures that every part of the process adds value for the customer by

transitioning smoothly from one value-adding stage to the next without unnecessary delays. Consequently, production is driven by actual demand, occurring just in time, at the precise location, and in the exact quantity required. As a further matter, Anderson (2010) explains that flow management focuses on efficiently completing work in progress by continuously monitoring and reporting each stage in the workflow, a process known as measuring flow. This helps that tasks move quickly and smoothly, thereby delivering timely value, reducing risks, and preventing delays and associated costs.

Some of the techniques used to manage flow are Kanban board, daily stand-up meetings, replenishment meetings, burn-down charts and lead time. Daily stand-up meetings in Kanban are somewhat different from the one in Scrum framework. The team goes through the board from right to left, starting from the last stage and moving toward the first stage in the workflow. This practice helps identify the bottlenecks and issues that disrupt the flow and prevent the team from completing tasks. Replenishment meetings are held regularly in order to review the backlog and prioritise the upcoming items for the purpose of filling the kanban system queue (Anderson, 2010). Lead time refers to the interval between the introduction of a work item into the system (doing stage) and its completion and exit from the system (done stage). It measures the total time required to process a work item from the beginning of the doing stage to finalising it.

Furthermore, there is a cumulative flow diagram (CFD) which represents one of the distinctive characteristics of the Kanban approach and is known as one of the most used diagrams. CFD visualizes the workflow and team's progress by showing the quantities of Work-in-Progress at each stage in the system (see Figure 13).

Figure 12: Cumulative Flow Diagram (CFD)



Source: Dash (2021).

5.6 Making process policies explicit

Process policies represent some sort of formal rules or standards established by an organization which need to be followed. They should be clear and simple, but also well-defined containing all necessary information, consistently applied, and easily changeable. Some of the examples are defining the criteria for pulling work items through phases (Definition of Done), defining guidelines for selecting new tasks when there is available capacity, categorizing levels of service, or setting a regular schedule for deliveries.

Kanban itself represents the set of principles and practices applied to an existing process, so that it can ensure a smooth and efficient workflow. Kanban is known for its flexibility and lack of rigid rules, but it follows two core principles: respecting the existing rules and modifying them when they become outdated. Sticking to the current process rules is necessary for identifying and addressing problems within the actual work process, but in order to make improvements, these rules should be periodically revised and modified.

5.7 Implementing fast feedback loops

Being agile means being adaptable and open to changes, modifying organizational processes to make them improve and evolve over time. Implementing fast feedback loops enables this evolutionary change by identifying where adjustments are needed and showing their areas of impact. The important thing in this process is to define the appropriate cadence of meetings and reviews, for the purpose of collecting proper and timely information. Klaus Leopold (2017) suggests three types of meetings as the most fitting ones for getting fast feedback and information exchange: standups, retrospectives and replenishment meetings.

5.8 Drive improvements based on methods and models

Kanban essentially serves as a method for continuous improvement. This means that changes are welcomed and by using adequate models teams are able to predict the impacts of changes. After implementing these changes, the data is analysed and compared to initial predictions to evaluate effectiveness. This procedure of observing results, proposing modifications, and comparing actual outcomes to predictions is as a typical example of the scientific method. Such an approach leads to learning at both individual and organizational levels and optimizing processes, which is the essence of Kanban (Damij, & Damij, 2021).

5.9 Advantages and disadvantages of Kanban framework

As a result of Kanban effectiveness, if we follow assessments like the 'State of Agile' reports, it's noticeable that there's been a significant surge in Kanban adoption. The statistics are showing a growth from 39% to 50% in 2016, following a further boost from 50% to 65% in 2017, as reported by VersionOne (2017, 2018). The main advantages achieved from the

implementation of Kanban methodologies include greater work visibility, increased transparency, facilitated communication, and enhanced control over workflow dynamics (Kirovska, & Koceski, 2015). Ahmad et al. (2013) disclose Kanban advantages as enhanced software delivery lead times, improved software quality, better communication and coordination, heightened delivery consistency, and reduced customer-reported defects.

What's more, Kanban provides numerous benefits across various teams. By analysing cycle times, identifying bottlenecks, and prioritizing tasks based on importance, teams can work more efficiently and effectively, ultimately leading to higher customer satisfaction and better project predictability (Adobe, 2025).

However, despite these benefits, practitioners also faced several challenges. Foremost among them was a deficiency in Kanban expertise, leading to struggles in navigating WIP (Work-In-Progress) limits. Another notable obstacle originates from the deep-rooted traditional culture within organizations, influencing the adoption and assimilation of Kanban methodologies (Kirovska & Koceski, 2015). Other authors state that the primary challenge is the lack of detailed guidelines for IT organizations on how to effectively use the Kanban approach. And additionally, there is a shortage of clear principles for introducing the different elements of Kanban (Al-Baik & Miller, 2015).

6 SCRUM AND KANBAN IN PRACTICE

6.1 Project description

Project Alpha was a digital platform created to connect beauty professionals, such as makeup artists, hairstylists, and others, with clients who were looking for their services. The platform was designed by a group of experienced English makeup professionals who brought experience, industry insights, and strong connections within the beauty sector. Initially launched as a startup, the project gained attention and recognition, and was eventually acquired by XZY, a leading British beauty company.

The platform was available through both mobile (compatible with iOS and Android) and web-based applications, offering users a seamless and accessible experience. Project Alpha was designed to serve two main user groups: professionals and customers. For beauty professionals, the platform provided tools to create their own digital salons where they could showcase their services, both on-site and off-site. This functionality helped professionals manage their offerings, attract clients, and simplify the booking process. For customers, Project Alpha delivered an intuitive interface for browsing services, booking appointments, and making secure payments. In addition to its core features, the platform included a variety of tools to improve the overall user experience, including a review system for feedback, refund management options, in-app messaging for smooth communication, and other functionalities that promoted convenience and trust.

In essence, Project Alpha functioned as a complete digital solution for the beauty industry, connecting the professional service providers and customers while using technology to make booking and service delivery easier. The project lasted 6 months and was managed using the Scrum framework. The team working on the project included four Developers, two QA engineers, a Designer, Product Owner and a Scrum Master.

6.2 Using Scrum

As mentioned earlier, the team chose the Scrum framework as the most suitable approach for managing the project. The work was organized into two-week Sprints, which is a common practice among software teams using Scrum.

Overview of implemented Scrum events:

- Daily stand-ups were held each morning and typically lasted around 15 minutes. The point of these meetings was to help the team align, share progress, and raise potential blockers.
- Sprint Planning sessions took place every second Monday at the beginning of each new Sprint. During these meetings, the entire team collaborated to define and estimate the workload. The Planning Poker technique, utilizing Fibonacci sequence, was used to assign story point estimates to user stories.
- Sprint Retrospectives were happening on the last Friday of each Sprint, giving the team the opportunity to reflect on the previous Sprint, talk about issues and identify areas for improvement.
- Backlog Refinement sessions were usually held on Wednesdays during the first week of the Sprint. These sessions included the Product Owner, Scrum Master, Lead Developer, and Designer, and focused on reviewing and updating the items in the product backlog.
- Sprint Reviews took place every Monday at the start of a new Sprint, during which the outcomes of the previous Sprint were demonstrated to the client.

During the project, the team used Atlassian Jira as the primary tool for task tracking, workflow organisation, and time management. For the purposes of this thesis, relevant data were exported to Microsoft Excel file. Tasks were progressing through defined columns on the board: To Do, In Progress, Testing, and Done. The screenshot of Sprint 1 is presented in the thesis (see Figure 14), while the full Microsoft Excel file is included in Appendix 1 (Scrum progress tracking).

Figure 13: Scrum Progress Tracking - Sprint 1

Sprint 1			
Sprint Backlog	Scrum Board Work in Progress	Testing	Done
Registration 2			
Email verification - FE 5			
Email verification - BE 2			
Web email verification 5			
Login 3			
Forgot password 5			

Sprint 1			
Sprint Backlog	Scrum Board Work in Progress	Testing	Done
	Email verification - FE 5		
Email verification - BE 2	Login 3	Registration 2	
Web email verification 5	Forgot password 5		

Sprint 1			
Sprint Backlog	Scrum Board Work in Progress	Testing	Done
	Email verification - FE 5		Registration 2
	Web email verification 5	Email verification - BE 2	
	Forgot password 5	Login 3	

Source: Own work

Task completion overview:

- In Sprint 1, five tasks were completed, one task was carried over to the next Sprint.
- In Sprint 2, five new tasks were planned in addition to one uncompleted task from the previous Sprint; however, only four new tasks were completed, plus the remaining task from Sprint 1.
- In Sprint 3, five new tasks were added along with one carryover task. By the end of the Sprint, four new tasks and the carryover task were completed. One new task was added mid-Sprint and completed, while another was postponed due to the need for further refinement.
- Sprint 4 involved four planned tasks, all of which were completed.
- In Sprint 5, all five planned tasks were successfully delivered.

Challenges encountered:

- Sprint 1 initiated issues for the subsequent Sprints, as not all planned tasks were completed on time.
- Sprint 2 created difficulties for the QA team, as several tasks accumulated toward the end of the Sprint, making it difficult to complete testing on time. See the whole process in the Appendix 1.

- The most significant issues arose in Sprint 3, where one poorly defined task was included despite having many unresolved questions. This caused changes in the Sprint scope, which later led to delays and created uncertainty in execution.

Overall, the main challenge during this project phase was the frequent carryover of unfinished tasks into subsequent Sprints. This was largely due to underestimated development efforts and unexpected complexities.

6.3. Using Kanban

In order to explore whether Kanban could offer improvements over Scrum, an ideal scenario was simulated in which the same project was re-evaluated using Kanban principles. These principles focus on continuous flow, clearly defined WIP limits, and visual task management.

The Kanban board was designed in Microsoft Excel and structured with the following workflow stages: Ready, Analyse (Doing and Done), Develop (Doing and Done), Test, Ready to Deploy, and Deployed. Selected screenshots showing how tasks progressed are presented below (see Figure 15). The complete multi-sheet Microsoft Excel file is provided in Appendix 2 (Kanban progress tracking) for full reference.

Figure 14: Kanban Progress Tracking - Day 1

Backlog	Ready	Analyse		Develop	
		Doing (2)	Done	Doing (4)	Done
	Login	Email verification - BE A: ● D: T: DD DR LT 1	Registration A: ● D: T: DD DR LT 1		
	Forgot password				
	Web email verification				
Business profile basics					
Add services and prices					
Upload photos					
Working hours					
Payment basics					
MOBILE CUSTOMER SETTINGS					
WEB CUSTOMER SETTINGS					
MOBILE BOOKING					
WEB BOOKING					

Source: Own work

In order to establish WIP limits, and in accordance with Anderson's (2010) recommendation, it was decided to begin with modest values in order to stabilize flow and identify potential bottlenecks early. Therefore, the starting point was to define WIP limits based on the workload distribution, while keeping the same team structure used in the Scrum framework, with the only exception of having a Project Manager replacing the roles of Product Owner and Scrum Master.

For the Analyse (Doing) phase, the WIP limit was set to 2 tasks per day, allowing the Designer, supported by the Project Manager, to manage a reasonable number of tasks simultaneously. In the Development (Doing) phase, one task per developer was estimated as reasonable, resulting in a WIP limit of 4, ensuring focused work without exceeding capacity. The Testing phase was limited to 3 tasks, matching the capacity of two QA engineers who can verify at least one task per day, while also providing enough buffer and flexibility to handle new tasks coming from development.

As mentioned earlier, these limits can be refined iteratively based on actual performance, which is consistent with Kanban's continuous improvement mindset. A detailed breakdown is provided in Appendix 2.

Figure 15: Kanban Progress Tracking - Day 2

Backlog	Ready	Analyse		Develop	
		Doing (2)	Done	Doing (4)	Done
Business profile basics Add services and prices Upload photos Working hours Payment basics MOBILE CUSTOMER SETTINGS WEB CUSTOMER SETTINGS MOBILE BOOKING WEB BOOKING		Forgot password A: ● D: ● T: ● DD DR LT 2	Login A: ● D: ● T: ● DD DR LT 2	Registration A: ● D: ● T: ● DD DR LT 1	
		Web email verification A: ● D: ● T: ● DD DR LT 2		Email verification - FE A: ● D: ● T: ● DD DR LT 1	
				Email verification - BE A: ● D: ● T: ● DD DR LT 1	

Source: Own work

In summary, tasks moved smoothly through the workflow (see Figure 16), and implementing WIP limits helped prevent bottlenecks, especially in the development and testing stages. The team successfully completed and deployed 7 tasks within 10 working days. To conclude, the

simulation suggests that the use of Kanban could've enabled the team to deliver work in smaller, more frequent batches, reducing task carryover and improving flow stability. On the other hand, while the scenario assumes an optimal implementation, more realistic adoption of Kanban may involve some challenges, especially in shifting from batch delivery to continuous flow and enforcing consistent WIP constraints.

7 SCRUM AND KANBAN COMPARISON

To see and evaluate how well Scrum and Kanban would work for Project Alpha, both frameworks were applied to the same set of tasks. Scrum was based on actual project data, while Kanban was tested in an idealized, flow-oriented scenario. The comparison below explores how each approach affected delivery efficiency, team workload, flexibility in the processes, and overall task management.

7.1 Task flow and delivery rhythm

Scrum followed fixed two-week Sprints, with around 4 to 5 tasks completed per Sprint. In the Kanban simulation, the team completed 7 tasks in 10 working days, showing a more responsive and continuous workflow. While Scrum's ensured regular release cycles with its time-boxed approach, it often delayed the deployment of ready tasks until the end of the Sprint. Kanban, on the other hand, allowed tasks to move forward as soon as they were completed in each phase, reducing lead times and increasing delivery frequency.

7.2 Work in progress (WIP) and workflow management

Scrum used Sprint Plannings and Story Points to estimate capacity. Even though this practice may work for Scrum teams and provide accurate estimations, in this case not having explicit WIP limits occasionally led to overcommitment. This was particularly obvious in Sprints 2 and 3, where testing bottlenecks and poorly defined tasks slowed the progress. In contrast, Kanban used clearly defined WIP limits: 2 for Analyse, 4 for Development, and 3 for Testing, based on the team's structure and workload. These modest limits helped maintain a steady pace, reduced bottlenecks, and supported continuous delivery by keeping tasks predictably moving through each phase.

7.3 Flexibility

Scrum provided structured planning cycles but limited flexibility once a Sprint began. While scope changes during the Sprint are generally discouraged, some exceptions were made to keep the team productive. In Sprint 3, for example, a poorly defined task was postponed due to unresolved uncertainties, and a smaller, properly prepared task was added mid-Sprint to avoid idle developer time. On the other hand, Kanban's pull-based system allowed the team

to adjust priorities and introduce new tasks without disrupting overall flow, as long as WIP limits were respected.

7.4 Team collaboration and coordination

Both methods supported collaboration and team work, though in different ways. Scrum's regular ceremonies (e.g., Daily meetings, Sprint Planning, Reviews, Retrospectives) ensured alignment and accountability, but also took considerable meeting time. Kanban made coordination easier by focusing on visualizing the workflow and tracking the progress through daily check-ins. Even though there are no regular ceremonies like in Scrum, Kanban was complemented by daily meetings and periodic retrospectives, to make sure there is a regular team communication and continuous improvement. It also supported cross-functional collaboration, allowing more flexibility between the roles and advocating for shared ownership of work.

7.5 Team motivation

An important advantage of Scrum, as highlighted in team feedback, was the sense of accomplishment experienced at the end of each Sprint. The structured cadence and visible completion of Sprint goals created a feeling of progress and increased the motivation, which was less noticeable in the continuous flow model of Kanban. Although Kanban improved efficiency, it lacked natural milestones, so teams had to put more effort to recognize achievements and keep the engagement.

7.6 Challenges and limitations

Scrum's main challenges were the carryovers of unfinished tasks and difficulties in estimating effort accurately, especially for complex or poorly defined tasks. Kanban addressed some of these issues by removing rigid time-boxes and using WIP limits to stabilize the flow. However, successful use of Kanban depends on team discipline, regular flow monitoring, and accepting the culture of continuous delivery. Also, these elements may require adjustments for teams more familiar with approaches based on iterations (e.g., Scrum).

7.7 Summary

Table 1 provides a summary and a side-by-side comparison of Scrum and Kanban methods applied on Project Alpha. The table highlights the main differences in the workflow structure, flexibility, task throughput, collaboration practices, and so on. The goal was to summarize the practical advantages and disadvantages observed for each method during the execution of the project.

Table 1: Scrum and Kanban Comparison

Characteristics	Scrum Framework	Kanban Framework
Workflow structure	Time-boxed Sprints (two weeks)	Continuous flow, no Sprints
Planning	Sprint Planning every two weeks	Ongoing planning
Flexibility	Limited - scope usually fixed during Sprint	High - tasks pulled if available capacity
Motivation and team spirit	Strong sense of accomplishment at the end of the Sprint	Less visible milestones; requires more effort to build motivation
Feedback and reflection	Regular Retrospective and Review meetings	Requires deliberate scheduling of feedback loops
Task flow	4–5 tasks completed per Sprint	7 tasks completed in two weeks
WIP management	Based on Sprint capacity expressed through Story Points	Explicit WIP limits (2 Analyse, 4 Develop, 3 Test)
Bottlenecks	Testing phase overloaded towards the end of the Sprint	Better control of bottlenecks through WIP limits
Tools	Jira board with columns: To Do, In Progress, Testing, Done	Jira board modified with WIP limits and different columns: Analyze (Doing/Done), Develop (Doing/Done), Test, Ready to Deploy, Deployed

Source: Adapted from Rehkopf (n.d.) and Dave (n.d.)

8 CONCLUSION

This master thesis explored how Scrum and Kanban, the most popular agile methods, work in practice, concretely in a software development project. The goal was to generate a comparative analysis based on the outcomes of both methods in order to provide recommendations and establish some guidelines for better usage of agile methods in IT project management.

Firstly, the research included a chronological overview of the project management development, from traditional methods to the agile approach, which focuses on customer

collaboration, quick and continuous delivery, and adapting to change. From there, it moved on to a closer look of Scrum and Kanban, the two frameworks that support agile values in very different ways. Scrum uses time-boxed Sprints and defined roles to provide structure and some rhythm to the process, while Kanban focuses more on visualizing work, limiting work in progress and managing flow to keep things moving smoothly.

In the empirical part of the research Scrum was applied to a real project, whose team was made up of developers, QA engineers, a designer, a Product Owner, and a Scrum Master. Over several two-week Sprints, the team followed the usual Scrum ceremonies like daily stand-ups, planning sessions, reviews and retrospectives. This method helped build a sense of alignment and shared purpose, however, there were some challenges like task spillover and testing delays, which showed that the structure of Scrum could sometimes get in the way, especially when estimates were wrong. In order to see how Kanban might perform differently, the same tasks were used in an ideal scenario with the format of continuous flow with clear WIP limits. The difference was noticeable: tasks moved through the workflow more quickly, and more work got done in the same amount of time. Setting WIP limits based on team capacity helped to avoid bottlenecks and made the process feel smoother. Nevertheless, the Kanban approach wasn't without its downsides. Without the regular feedback and Sprint milestones, some team members missed the sense of progress and completion that Scrum naturally creates. This points to a conclusion that while Kanban can improve efficiency and flexibility, it may require extra effort to keep the team motivated and in harmony.

To summarise, both Scrum and Kanban methods bring valuable strengths to the table. Scrum is great for structure, focus, and team rhythm. Kanban on the other hand, is better when it comes to flow, flexibility and quick changes. Which one works best depends on the nature of the work, the team's preferences, and the goals of the organization. It is important to emphasize that the better results of the Kanban approach, obtained through simulation under ideal conditions, serve more as a recommendation, which does not mean that the same results would be achieved in the real world. Ultimately, choosing the right approach comes down to understanding the team's needs, being adaptable, and constantly looking for ways to improve.

REFERENCE LIST

1. Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002) Agile software development methods: Review and analysis. *VTT publication 478, Espoo, Finland, 107p*. <https://doi.org/10.48550/arXiv.1709.08439>
2. Adobe (2025). *Kanban — a complete beginner's guide*. Adobe for Business. <https://business.adobe.com/blog/basics/what-is-kanban?>

3. Ahmad, M. O., Dennehy, D., Conboy, K., & Oivo, M. (2018). Kanban in Software Engineering: A Systematic Mapping Study. *Journal of Systems and Software*, 137, 96-113. <https://doi.org/10.1016/j.jss.2017.11.045>
4. Ahmad, M.O., Markkula, J., & Oivo, M. (2013). Kanban in software development: A systematic literature review. *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, 9-16. <https://doi.org/10.1109/SEAA.2013.28>
5. Alam, I., Sarwar, N., & Noreen, I. (2022) Statistical analysis of software development models by six-pointed star framework. *PLoS One*, 17(4). <https://doi.org/10.1371/journal.pone.0264420>
6. Al-Baik, O., & Miller, J. (2015) The kanban approach, between agility and leanness: a systematic review. *Empirical Software Engineering*, 20, 1861–1897. <https://doi.org/10.1007/s10664-014-9340-x>
7. Alqudah, M.K., & Razali, R. (2017). A comparison of scrum and Kanban for identifying their selection factors. *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*, 1-6. doi: 10.1109/ICEEI.2017.8312434
8. Anderson, D. J. (2010). Kanban-Successful Evolutionary Change for Your Technology Business. *Blue Hole Press*. ISBN: 978-0-9845214-2-9.
9. Arora R., & Arora N. (2016) Analysis of SDLC Models. *International Journal of Current Engineering and Technology (IJCET)*, 6(1), 268-272. <https://inpressco.com/wp-content/uploads/2016/02/Paper48268-272.pdf>
10. Asproni, G. (2006). An Introduction to Scrum. *Software Developer's Journal*, 6(1). https://www.asprotunity.com/resources/articles/AnIntroductionToScrum_SDJ_06-2006_EN.pdf
11. Atrey, A. (2023). *Top 5 risks of scrum methodology*. LinkedIn. <https://www.linkedin.com/pulse/top-5-risks-scrum-methodology-anuj-atrey/>
12. Bassil, Y. (2011). A Simulation Model for the Waterfall Software Development Life Cycle. *International Journal of Engineering & Technology (iJET)*, 2(5). <https://arxiv.org/pdf/1205.6904>
13. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Manifesto for Agile Software Development. *Agile Alliance*. <https://agilemanifesto.org>
14. Boehm, B.W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5), 61-72. <https://doi.org/10.1109/2.59>
15. Brechner, E. (2015). Agile Project Management with Kanban. *Microsoft Press*. ISBN: 978-0-7356-9895-6
16. Brewer, J. L., & Dittman, K. C. (2018). Methods of IT Project Management: Third Edition. *Purdue University Press*. <https://doi.org/10.2307/j.ctv15wxrd5>
17. Cathy. (2020). *The Kanban method in IT development projects*. Bocasy.com

18. Cervone, F. H. (2011). Understanding agile project management methods using Scrum. *OCLC Systems & Services: International digital library perspectives*, 27(1). <https://doi.org/10.1108/10650751111106528>
19. Ciric, D., Lalic, B., Gracanin, D., Tasic, N., Delic, M., & Medic, N. (2019). Agile vs. Traditional Approach in Project Management: Strategies, Challenges and Reasons to Introduce Agile. *Procedia Manufacturing*, 39. <https://doi.org/10.1016/j.promfg.2020.01.314>
20. Colucci, L. (2016). Case study of a WIP limit implementation: Why, when and how to use WIP limits. *Platformatec Blog*. <https://blog.plataformatec.com.br/2016/09/case-study-of-a-wip-limit-implementation-why-when-and-how-to-use-wip-limits/>
21. Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research*, 20(3), 329-354. <https://doi.org/10.1287/isre.1090.0236>
22. Cooper, R.G., & Sommer, A.F. (2016). The Agile–Stage-Gate Hybrid Model: A Promising New Approach and a New Research Opportunity. *Journal of Product Innovation Management*, 33(5), 513–526. <https://doi.org/10.1111/jpim.12314>
23. Coram, M., & Bohner, S. (2005). The impact of agile methods on software project management. *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS '05)*, 363–370. <https://doi.org/10.1109/ECBS.2005.68>
24. Corona, E., & Pani, F. E. (2013). A Review of Lean-Kanban Approaches in the Software Development. *WSEAS Transactions on Information Science and Applications*, 10(1), 1–13.
25. Damij, N., & Damij, T. (2021). An Approach to Optimizing Kanban Board Workflow and Shortening the Project Management Plan. *IEEE Transactions on Engineering Management*, 71. <https://doi.org/10.1109/TEM.2021.3120984>
26. Dash, P. (2021). *Predicting Impediments Using Cumulative Flow Diagram (CFD)*. Medium.com
27. Dave, P. (n.d.). *Scrum VS Kanban*. Scrum Alliance. <https://resources.scrumalliance.org/Article/scrums-vs-kanban>
28. Derby, E., & Larsen, D. (2006). *Agile Retrospectives: Making Good Teams Great*. The Pragmatic Bookshelf. <https://pragprog.com/titles/dlret/agile-retrospectives/>
29. Digital. (2022). *The 16th Annual State of Agile Report*. Digital.ai. <https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf>
30. Dora, S. K., & Dubey, P. (2013). Software development life cycle (SDLC) analytical comparison and survey on traditional and agile methodology. *National Monthly Refereed Journal of Research in Science & Technology*, 2(8), 22–30.
31. Duncan, W. R. (1993). The process of project management. *Project Management Journal*, 24(3), 5-10.

32. Eby, K. (2022). *How to Identify and Manage Constraints in Project Management*. Smartsheet.com <https://www.smartsheet.com/content/project-constraints>
33. Fowler, F. (2019). *Navigating Hybrid Scrum Environments: Understanding the Essentials, Avoiding the Pitfalls*. Apress Berkeley. <https://doi.org/10.1007/978-1-4842-4164-6>
34. Fowler, M., & Highsmith, J. (2001). The Agile manifesto. *Software Development*, 9(8), 28-32.
35. Ganev, P. (2018). Advantages and disadvantages of using Scrum, Kanban and Scrumban for software development. https://www.academia.edu/36983292/Peter_Ganev_Advantages_and_disadvantages_of_using_Scrum_Kanban_and_Scrumban_for_software_development
36. Gupta, A., Rawal, A. & Barge, Y. (2021). Comparative Study of Different SDLC Models. *International Journal for Research in Applied Science and Engineering Technology*, 9(11). <https://doi.org/10.22214/ijraset.2021.38736>
37. Hass, K. B. (2007). The blending of traditional and agile project management. *PM world today*, 9(5), 1-8.
38. Heagney, J. (2011). *Fundamentals of Project Management 4th Edition*. AMACOM.
39. Hennigan, L. & Bottorff, C. (2024). *What Is A Kanban Board? The Ultimate Guide*. Forbes Advisor. <https://www.forbes.com/advisor/business/software/what-is-kanban-board/>
40. Highsmith, J. (2009). *Agile project management: Creating innovative products* (2nd ed.). Pearson education.
41. Hughes, B., Ireland, R., West, B., Smith, N., & Shepherd, D. I. (2019). *Project Management for IT-Related Projects* (3rd ed.). BCS Learning & Development Limited.
42. Ikonen, M., Pirinen, E., Fagerholm, F., Kettunen, P., & Abrahamsson, P. (2011). On the impact of Kanban on software project work: An empirical case study investigation. In *2011 16th IEEE international conference on engineering of complex computer systems* (pp. 305-314). IEEE. <https://doi.org/10.1109/ICECCS.2011.37>
43. Jeffries, R. (2001). *Essential XP: Card, Conversation, Confirmation*. <https://ronjeffries.com/xprog/articles/expcardconversationconfirmation/>
44. Kelley, J. E., Walker, M. R., & Sayer, J. S. (1989). The origins of CPM: a personal history. *PM Network*, 3(2), 7–22.
45. Kerzner, H. (2009). *Project management: a systems approach to planning, scheduling, and controlling* (10th ed.). John Wiley & Sons, Inc.
46. Kirovska, N., & Koceski, S. (2015). Usage of Kanban methodology at software development teams. *Journal of Applied Economics and Business*, 3(3), 25–34.
47. Kozak-Holland, M. (2011). Building the case for historical project management. *10th International Research Network on Organizing by Projects Conference (IRNOP)*, Publication No. 7.

48. Kute S. S., & Thorat, S. D. (2014) A review on various software development life cycle (SDLC) models. *International Journal of Research in Computer and Communication Technology*, 3(7), 778-779.
49. Lei, H., Ganjeizadeh, F., Jayachandran, P. K., & Ozcan, P. (2017). A statistical analysis of the effects of Scrum and Kanban on software development projects. *Robotics and Computer-Integrated Manufacturing*, 43, 59–67. <https://doi.org/10.1016/j.rcim.2015.12.001>
50. Leopold, K. (2017). *Practical Kanban: From Team Focus to Creating Value*. LEANability.
51. Lock, D. (2014). *The Essentials of Project Management* (4th ed.). Routledge. <https://doi.org/10.4324/9781315239941>
52. Lutwama, P., Dlulane, M., Pillay, T., Hassan, F.S., & Grobbelaar, S. (2024). Agile: advantages, disadvantages, enablers, and barriers. *South African Journal of Industrial Engineering*, 35(4), 66-76. <https://doi.org/10.7166/35-4-3058>
53. Mackay, H., Carne, C., Beynon-Davies, P., & Tudhope, D. (2000). Reconfiguring the user: Using rapid application development. *Social studies of science*, 30(5), 737-757. <https://doi.org/10.1177/030631200030005004>
54. Mahnič, V. (2015). From Scrum to Kanban: introducing lean principles to a software engineering capstone course. *International Journal of Engineering Education*, 31(4), 1106–1116.
55. Makadia, H. (n.d.) *Your complete guide to rapid application development (RAD)*. Maruti Techlabs. <https://marutitech.com/rapid-application-development/>
56. Martin, J. (1991). *Rapid application development*. Macmillan Publishing Company.
57. Mishra, A., & Dubey, D. (2013). Suitability analysis of various software development life cycle models. *International Journal of Electronics Communication and Computer Engineering*, 4(6), 98-101.
58. Motion. (2023). *How to Use a Burndown Chart: Everything You Need to Know*. Motion.
59. Nakkasem, T. (2020) *V-Model: An Improvement of Waterfall*. Medium.
60. Ohno, T. (1988). *Toyota Production System: Beyond Large-Scale Production*. CRC Press.
61. Olorunshola, O. E., & Ogwueleka, F. N. (2022). Review of system development life cycle (SDLC) models for effective application delivery. In Joshi, A., Mahmud, M., Ragel, R.G., Thakur, N.V. (eds), *Information and Communication Technology for Competitive Strategies (ICTCS 2020). Lecture Notes in Networks and Systems*, 191. Springer Singapore. https://doi.org/10.1007/978-981-16-0739-4_28
62. Pargaonkar, S. (2023). A comprehensive research analysis of software development life cycle (SDLC) agile & waterfall model advantages, disadvantages, and application suitability in software quality engineering. *International Journal of Scientific and Research Publications (IJSRP)*, 13(08), 120-124. <https://doi.org/10.29322/IJSRP.13.08.2023.p14015>

63. Project Management Institute. (2008). *PMBOK® Guide*. Project Management Institute.
64. Project Management Institute. (2013). *PMBOK® Guide*. Project Management Institute.
65. Project Management Institute. (2021). *PMBOK® Guide*. Project Management Institute.
66. Project Management Institute. (n.d.). *Product Owner*. Project Management Institute.
<https://www.pmi.org/disciplined-agile/product-owner>
67. Radigan, D. (n.d.). *Story points and estimation*. Atlassian.
<https://www.atlassian.com/agile/project-management/estimation>
68. Rehkopf, M. (n.d.). *Kanban vs. scrum: which agile are you?* Atlassian.
<https://www.atlassian.com/agile/kanban/kanban-vs-scrum>
69. Rehkopf, M. (n.d.). *User Stories with examples and a template*. Atlassian.
<https://www.atlassian.com/agile/project-management/user-stories>
70. Royce, W.W. (1970). *Managing the development of large software systems*. *IEEE WESCON*, 1–9. <https://doi.org/10.7551/mitpress/12274.003.0035>
71. Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular agile process*. Addison-Wesley Professional.
72. Salameh, H. (2014). What, when, why, and how? A comparison between agile project management and traditional project management methods. *International Journal of Business and Management Review*, 2(5), 52-74.
73. Schwaber, K., & Beedle, M. (2001). *Agile software development with Scrum*. Prentice Hall PTR.
74. Schwaber, K., & Sutherland, J. (2011). *The Scrum Guide. The definitive guide to scrum: The rules of the game*. Scrum.org
https://res.cloudinary.com/mitchlacey/image/upload/v1589750998/Scrum_Guide_October_2011_ilgx9c.pdf
75. Schwaber, K., & Sutherland, J. (2017). *The Scrum Guide: The definitive guide to scrum: The rules of the game*. Scrum.org
<https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf>
76. Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org
<https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf>
77. Schwalbe, K. (2017). *An introduction to project management* (6th ed.). Schwalbe Publishing. ISBN 978-1544701899.
78. Scrum Alliance (n.d.). *What is a Sprint Review?* Scrum Alliance.
<https://resources.scrumalliance.org/Article/sprint-review>
79. Scrum Institute (n.d.). *Scrum team and Scrum roles in agile project management*. Scrum Institute.
https://www.scrum-institute.org/Scrum_Roles_The_Scrum_Team.php
80. Scrum.org. (n.d.). *What is a Daily Scrum?* Scrum.org.
<https://www.scrum.org/resources/what-is-a-daily-scrum>

81. Scrum.org. (n.d.). *What is Scrum?* Scrum.org.
<https://www.scrum.org/resources/what-scrum-module>
82. Seymour, T., & Hussein, S. (2014). The History Of Project Management. *International Journal of Management & Information Systems (IJMIS)*, 18(4), 233-240. <https://doi.org/10.19030/ijmis.v18i4.8820>
83. Sharma, S., Sarkar, D., & Gupta, D. (2012). *Agile processes and methodologies: A conceptual study*. *International Journal on Computer Science and Engineering*, 4(5), 892-898.
84. Silva, A., Dilorenzo, E., Perkusich, M., Santos, D., Almeida, H., & Perkusich, A. (2018). Definition of Done from academy to the industry: An exploratory survey. *International Journal of Engineering Trends and Technology (IJETT)*, 58(2), 72–78. <https://doi.org/10.14445/22315381/IJETT-V58P214>
85. Stacey, R. D., & Mowles, C. (2016). *Strategic management and organisational dynamics: The challenge of complexity to ways of thinking about organisations* (7th ed.). Pearson education.
86. Stretton, A. (2023). A Short History of Modern Project Management. *PM World Journal*, 12(7), Article 131.
87. Sugimori, Y., Kusunoki, K., Cho, F., & Uchikawa, S. (1977). Toyota production system and Kanban system Materialization of just-in-time and respect-for-human system. *International Journal of Production Research*, 15(6), 553–564. <https://doi.org/10.1080/00207547708943149>
88. Takeuchi, H., & Nonaka, I. (1986). The new product development game. *Harvard Business Review*, 64(1), 137–146.
89. Toyota Blog. (2013). *Kanban – Toyota Production System guide*. Toyota UK Magazine. <https://mag.toyota.co.uk/kanban-toyota-production-system/>
90. Trojanowska, J., & Dostatni, E. (2017). Application of the theory of constraints for project management. *Management and Production Engineering Review*, 8(3), 87-95. DOI: 10.1515/mpcr-2017-0031
91. Uludağ, Ö., Harders, N. M., & Matthes, F. (2019, July). Documenting recurring concerns and patterns in large-scale agile development. In *Proceedings of the 24th European Conference on Pattern Languages of Programs*, Article 27, 1–17. Association for Computing Machinery. <https://doi.org/10.1145/3361149.3361176>
92. VersionOne (2017). *The 11th Annual State of Agile Report*. VersionOne. <https://info.digital.ai/rs/981-LQX-968/images/SOA11.pdf>
93. VersionOne (2018). *The 12th Annual State of Agile Report*. VersionOne. <https://info.digital.ai/rs/981-LQX-968/images/SOA12.pdf>
94. Vijay Anand, R., & Dinakaran, M. (2016). Popular agile methods in software development: review and analysis. *International Journal of Applied Engineering Research*, 11(5), 3433-3437.
95. Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011). The agile requirements refinery: Applying SCRUM principles to software product

- management. *Information and Software Technology*, 53(1), 58-70.
<https://doi.org/10.1016/j.infsof.2010.08.004>
96. Wranek, A. (2023). *Look at Scrum Events and Meetings*. Experte.
<https://www.experte.com/project-management/scrum-events>

APPENDICES

Appendix 1: Summary in Slovene Language

Magistrsko delo raziskuje, kako Scrum in Kanban, dve najbolj priljubljeni agilni metodi v razvoju programske opreme, delujeta v praksi. Glavni cilj je bil primerjati ti dve metodi na konkretnem projektu ter na podlagi rezultatov podati priporočila za učinkovitejšo uporabo agilnih pristopov v IT-projektih.

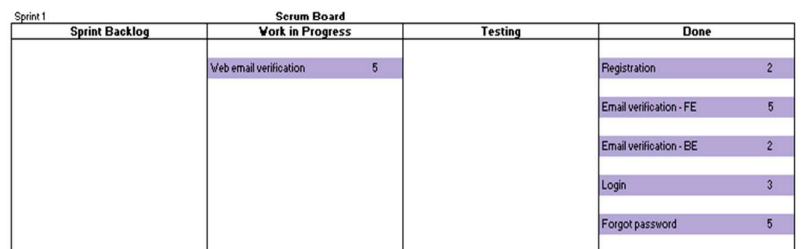
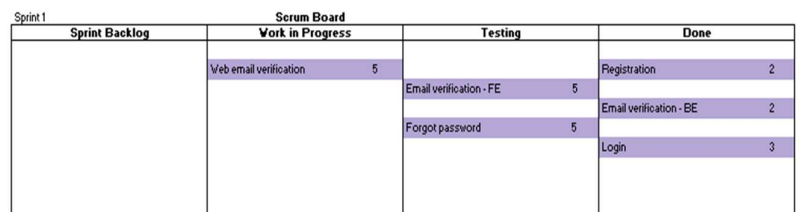
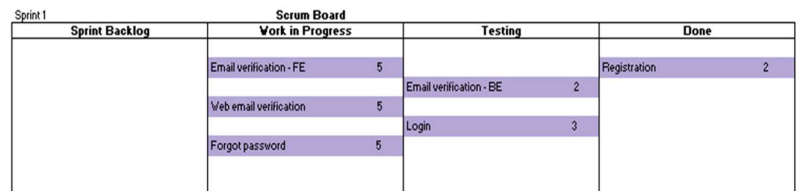
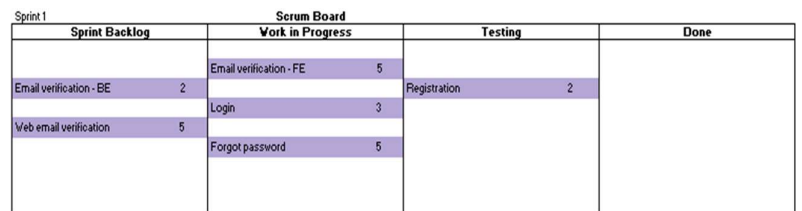
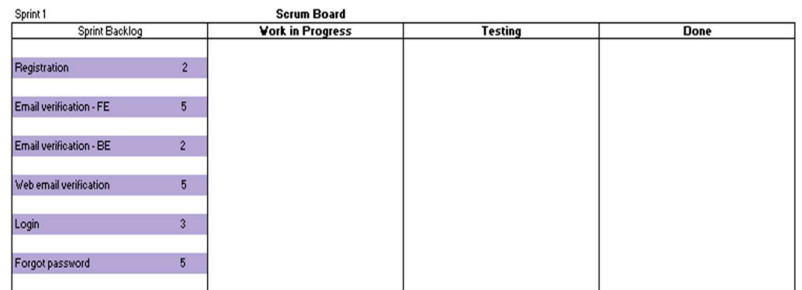
V teoretičnem delu je predstavljen razvoj projektnega vodenja – od klasičnih, tradicionalnih metod do agilnega pristopa, ki daje večji poudarek sodelovanju s strankami, hitri dostavi in prilagodljivosti. Sledita podroben opis Scruma in Kanbana ter primerjava njunih načinov dela, vlog v ekipi, organizacije nalog in ključnih pravil.

V praktičnem delu sta bili obe metodi uporabljeni na istem projektnem primeru (Projekt Alpha). Scrum je bil implementiran z dejansko ekipo (razvijalci, preizkuševalci, oblikovalec, lastnik produkta in Scrum mojster) skozi več dvotedenskih sprintov. Nato je bila simulirana uporaba Kanbana, pri čemer so bile uporabljene iste naloge z uvedbo omejitev dela v teku (t. i. WIP limits).

Na koncu sta obe metodi pokazali jasne prednosti, pa tudi določene pomanjkljivosti, kar je v delu podrobno analizirano. Ugotovitve predlagajo, da bi morale ekipe ostati prilagodljive, redno preverjati učinkovitost svojih procesov in izbrati pristop, ki najbolj ustreza njihovem načinu dela in potrebam posameznega projekta.

Sprint 1

User Stories	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
Registration	2	1								
Email verification - FE	5	3	2	2	2	1				
Email verification - BE	2	2	2	2	1					
Web email verification	5	5	5	3	3	2	2	1	1	1
Login	3	2	2	1	1					
Forgot password	5	5	3	3	2	2	2	2	1	
Total	22	18	14	11	9	5	4	3	2	1



Sprint 2

Story points

User Stories	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
Web email verification	1									
Business profile basics	2	2	2	1	1	1	1	1		
Upload photos	3	3	3	2	3	2	1	1	1	
Add services and prices	5	3	2	1						
Working hours	2	2	2	2	2	2	2	2	1	
Payment basics	8	8	8	8	8	8	5	3	3	3
Total	21	18	17	14	14	13	9	7	5	3

Backlog - Product plan	
Registration	2
Email verification	11
Login	3
Forgot password	5
21 sp	
Web email verification	1
Business profile basics	2
Upload photos	3
Add services and prices	5
Working hours	2
Payment basics	8
21 sp	
Customer settings - Main	2
Customer settings - My Addresses	3
Customer settings - Booking History	8
Customer Settings - Get Help & Legals	5
Customer Settings - Notifications	2
20 sp	
Web customer settings - My Addresses	5
Web customer settings - Password	2
Web customer settings - Main	8
Web customer Settings - Notifications	3
18 sp	
Booking - Time slot	2
Selecting and adding an address	3
Working hours and service location(s)	3
Booking - Payment	5
Booking details - Summary	8
21 sp	
WEB BOOKING	

Sprint 2		Scrum Board		
Sprint Backlog		Work in Progress	Testing	Done
Web email verification		1		
Business profile basics		2		
Upload photos		3		
Add services and prices		5		
Working hours		2		
Payment basics		8		

Sprint 2		Scrum Board		
Sprint Backlog		Work in Progress	Testing	Done
Upload photos		3		
Working hours		2		
Payment basics		8		
		Business profile basics	2	Web email verification 1
		Add services and prices	5	

Sprint 2		Scrum Board		
Sprint Backlog		Work in Progress	Testing	Done
		Upload photos	3	Web email verification 1
		Working hours	2	
		Payment basics	8	
			Business profile basics	2
			Add services and prices	5

Sprint 2		Scrum Board		
Sprint Backlog		Work in Progress	Testing	Done
		Upload photos	3	
		Working hours	2	Web email verification 1
		Payment basics	8	Add services and prices 5

Sprint 2		Scrum Board		
Sprint Backlog		Work in Progress	Testing	Done
		Payment basics	8	Web email verification 1
			Business profile basics	2
			Upload photos	3
			Working hours	2
			Add services and prices	5

Sprint 2		Scrum Board		
Sprint Backlog		Work in Progress	Testing	Done
		Payment basics	8	Business profile basics 2
				Upload photos 3
				Add services and prices 5
				Working hours 2
				Web email verification 1

Sprint 3

Story points

User Stories	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
Payment basics	3	2	1							
Customer settings - Main	2	1								
Customer settings - My Addresses	3	3	2	1						
Customer settings - Booking History	8	5	5	8	8	8	8	8	8	8
Customer settings - Get Help & Legals	5	5	5	3	3	2	1			
Customer settings - Notifications	2	2	2	2	2	2	2	2	1	
Customer Settings - Personal Info				1						
Total	23	18	15	15	13	12	11	10	9	8

Needs more planning

Backlog - Product plan		
Sprint 1	Registration	2
	Email verification	11
	Login	3
	Forgot password	5
		21 sp
Sprint 2	Web email verification	1
	Business profile basics	2
	Upload photos	3
	Add services and prices	5
	Working hours	2
	Payment basics	5
		18 sp
Sprint 3	Payment basics	3
	Customer settings - Main	2
	Customer settings - My Addresses	3
	Customer settings - Booking History	8
	Customer Settings - Get Help & Legals	5
	Customer Settings - Notifications	2
		23 sp
Sprint 4	Web customer settings - My Addresses	5
	Web customer settings - Password	2
	Web customer settings - Main	8
	Web customer Settings - Notifications	3
		18 sp
Sprint 5	Booking - Time slot	2
	Selecting and adding an address	3
	Working hours and service location(s)	3
	Booking - Payment	5
	Booking details - Summary	8
		21 sp
WEB BOOKING		

Sprint 3 Scrum Board			
Sprint Backlog	Work in Progress	Testing	Done
Payment basics 3			
Customer settings - Main 2			
Customer settings - My Addresses 3			
Customer settings - Booking History 8			
Customer Settings - Get Help & Legals 5			
Customer Settings - Notifications 2			

Sprint 3 Scrum Board			
Sprint Backlog	Work in Progress	Testing	Done
Customer Settings - Get Help & Legals 5	Payment basics 3		
Customer Settings - Notifications 2	Customer settings - My Addresses 3	Customer settings - Main 2	
	Customer settings - Booking History 8		

Sprint 3 Scrum Board			
Sprint Backlog	Work in Progress	Testing	Done
	Customer settings - Booking History 8		Customer settings - Main 2
	Customer Settings - Get Help & Legals 5	Payment basics 3	
	Customer Settings - Notifications 2	Customer settings - My Addresses 3	
		Customer Settings - Personal Info 1	

Sprint 3 Scrum Board			
Sprint Backlog	Work in Progress	Testing	Done
	Customer settings - Booking History 8		Customer settings - Main 2
	Customer Settings - Notifications 2	Customer Settings - Get Help & Legals 5	Payment basics 3
			Customer settings - My Addresses 3
			Customer Settings - Personal Info 1

Sprint 3 Scrum Board			
Sprint Backlog	Work in Progress	Testing	Done
	Customer settings - Booking History 8		Payment basics 3
			Customer settings - Main 2
			Customer settings - My Addresses 3
			Customer Settings - Get Help & Legals 5
			Customer Settings - Notifications 2
			Customer Settings - Personal Info 1

Sprint 4

Story points

User Stories	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
Web customer settings - My Addresses	5	5	5	3	3	2	1			
Web customer settings - Password	2	2	1							
Web customer settings - Main	8	8	8	5	5	3	3	2	1	
Web customer Settings - Notifications	3	3	3	3	3	2	1	1		
Total	18	18	17	11	11	7	5	3	1	0

Backlog - Product plan	
Registration	2
Email verification	11
Login	3
Forgot password	5
21 sp	
Web email verification	1
Business profile basics	2
Upload photos	3
Add services and prices	5
Working hours	2
Payment basics	5
18 sp	
Payment basics	3
Customer settings - Main	2
Customer settings - My Addresses	3
Customer Settings - Get Help & Legals	5
Customer Settings - Notifications	2
15sp	
Web customer settings - My Addresses	5
Web customer settings - Password	2
Web customer settings - Main	8
Web customer Settings - Notifications	3
18 sp	
Booking - Time slot	2
Selecting and adding an address	3
Working hours and service location(s)	3
Booking - Payment	5
Booking details - Summary	8
21 sp	
WEB BOOKING	

Sprint 4		Scrum Board			
Sprint Backlog		Work in Progress		Testing	Done
Web customer settings - My Addresses		5			
Web customer settings - Password		2			
Web customer settings - Main		8			
Web customer Settings - Notifications		3			

Sprint 4		Scrum Board			
Sprint Backlog		Work in Progress		Testing	Done
Web customer settings - My Addresses		5			
Web customer settings - Password		2			
Web customer Settings - Notifications		3			

Sprint 4		Scrum Board			
Sprint Backlog		Work in Progress		Testing	Done
Web customer Settings - Notifications		3			Web customer settings - Password 2
Web customer settings - My Addresses		5			
Web customer settings - Main		8			

Sprint 4		Scrum Board			
Sprint Backlog		Work in Progress		Testing	Done
Web customer settings - Main		8			Web customer settings - Password 2
Web customer Settings - Notifications		3			Web customer settings - My Addresses 5

Sprint 4		Scrum Board			
Sprint Backlog		Work in Progress		Testing	Done
Web customer settings - Password		2			
Web customer settings - My Addresses		5			
Web customer Settings - Notifications		3			
Web customer settings - Main		8			

Sprint 5

Story points

User Stories	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
Booking - Time slot	2	1	1	1	0.5					
Selecting and adding an address	3	2	1							
Working hours and service location(s)	3	3	3	3	3	3	1			
Booking - Payment	5	5	5	5	5	3	3	2	1	
Booking details - Summary	8	8	5	5	5	3	2	2	1	
Total	21	19	15	14	13.5	9	6	4	2	0

Backlog - Product plan		
Sprint 1	Registration	2
	Email verification	11
	Login	3
	Forgot password	5
Sprint 2	Web email verification	1
	Business profile basics	2
	Upload photos	3
	Add services and prices	5
	Working hours	2
Sprint 3	Payment basics	5
	Payment basics	3
	Customer settings - Main	2
	Customer settings - My Addresses	3
	Customer Settings - Get Help & Legals	5
Sprint 4	Customer Settings - Notifications	2
	Web customer settings - My Addresses	5
	Web customer settings - Password	2
	Web customer settings - Main	8
	Web customer Settings - Notifications	3
Sprint 5	Booking - Time slot	2
	Selecting and adding an address	3
	Working hours and service location(s)	3
	Booking - Payment	5
	Booking details - Summary	8
WEB BOOKING		

Sprint 5 Scrum Board			
Sprint Backlog	Work in Progress	Testing	Done
Booking - Time slot 2			
Selecting and adding an address 3			
Working hours and service location(s) 3			
Booking - Payment 5			
Booking details - Summary 8			

Sprint 5 Scrum Board			
Sprint Backlog	Work in Progress	Testing	Done
	Selecting and adding an address 3		
Working hours and service location(s) 3		Booking - Time slot 2	
Booking - Payment 5	Booking details - Summary 8		

Sprint 5 Scrum Board			
Sprint Backlog	Work in Progress	Testing	Done
	Booking details - Summary 8		
	Working hours and service location(s) 3	Booking - Time slot 2	
	Booking - Payment 5	Selecting and adding an address 3	

Sprint 5 Scrum Board			
Sprint Backlog	Work in Progress	Testing	Done
	Booking details - Summary 8		Booking - Time slot 2
	Booking - Payment 5	Working hours and service location(s) 3	Selecting and adding an address 3

Sprint 5 Scrum Board			
Sprint Backlog	Work in Progress	Testing	Done
			Booking - Time slot 2
			Selecting and adding an address 3
			Working hours and service location(s) 3
			Booking - Payment 5
			Booking details - Summary 8

Appendix 3: Kanban progress tracking

Backlog

Backlog
AUTHENTICATION
SUPPLIER ONBOARDING
MOBILE CUSTOMER SETTINGS
WEB CUSTOMER SETTINGS
MOBILE BOOKING
WEB BOOKING

Day 1

Backlog	Ready	Analyze		Develop		Test (3)	Ready to Deploy (4)	Deployed
		Doing (2)	Done	Doing (4)	Done			
	Login	Email verification - BE	Registration					
	Forgot password	A: ● D: T:	A: ● D: T:					
	Web email verification	DD DR LT 1	DD DR LT 1					
Business profile basics								
Add services and prices								
Upload photos								
Working hours								
Payment basics								
MOBILE CUSTOMER SETTINGS								
WEB CUSTOMER SETTINGS								
MOBILE BOOKING								
WEB BOOKING								

Day 2

Backlog	Ready	Analyze		Develop		Test (3)	Ready to Deploy (4)	Deployed
		Doing (2)	Done	Doing (4)	Done			
	Business profile basics	Forgot password	Login	Registration				
	Add services and prices	A: ● D: T: DD DR LT 2	A: ● D: T: DD DR LT 2	A: ● D: ● T: DD DR LT 1				
Upload photos		Web email verification		Email verification - FE				
Working hours		A: ● D: T: DD DR LT 2		A: ● D: ● T: DD DR LT 1				
Payment basics								
MOBILE CUSTOMER SETTINGS				Email verification -BE				
WEB CUSTOMER SETTINGS				A: ● D: ● T: DD DR LT 1				
MOBILE BOOKING								
WEB BOOKING								

Day 3

Backlog	Ready	Analyze		Develop		Test (3)	Ready to Deploy (4)	Deployed
		Doing (2)	Done	Doing (4)	Done			
	Add services and prices	Web email verification A: ●● D: T: DD DR LT 2	Forgot password A: ●● D: T: DD DR LT 2	Registration A: ● D: ●● T: DD DR LT 1				
	Upload photos							
Working hours		Business profile basics A: ● D: T: DD DR LT 3		Email verification - FE A: ● D: ●● T: DD DR LT 1				
Payment basics								
MOBILE CUSTOMER SETTINGS				Email verification - BE A: ● D: ●● T: DD DR LT 1				
WEB CUSTOMER SETTINGS								
MOBILE BOOKING				Login A: ● D: ● T: DD DR LT 2				
WEB BOOKING								

Day 4

Backlog	Ready	Analyze		Develop		Test (3)	Ready to Deploy (4)	Deployed
		Doing (2)	Done	Doing (4)	Done			
	Working hours Payment basics	Add services and prices A: ● D: T: DD DR LT 4 Upload photos A: ● D: T: DD DR LT 4	Business profile basics A: ●● D: T: DD DR LT 3	Email verification - FE A: ● D: ●●● T: DD DR LT 1 Forgot password A: ●● D: ● T: DD DR LT 2 Web email verification A: ●● D: ● T: DD DR LT 2	Login A: ● D: ●● T: DD DR LT 2 Email verification - BE A: ●● D: ●●● T: DD DR LT 1	Registration A: ● D: ●● T: ● DD DR LT 1		
Customer settings - Main								
Customer settings - My Addresses								
Customer settings - Booking History								
Customer Settings - Get Help & Legals								
Customer Settings - Notifications								
WEB CUSTOMER SETTINGS								
MOBILE BOOKING								
WEB BOOKING								

Day 5

Backlog	Ready	Analyze		Develop		Test (3)	Ready to Deploy (4)	Deployed
		Doing (2)	Done	Doing (4)	Done			
Working hours Payment basics Customer settings - Main Customer settings - My Addresses Customer settings - Booking History Customer Settings - Get Help & Legals Customer Settings - Notifications WEB CUSTOMER SETTINGS MOBILE BOOKING WEB BOOKING	Working hours	Upload photos	Add services and prices	Forgot password	Email verification - FE	Registration		
		A: ●● D: T:	A: ●● D: T:	A: ●● D: ●● T:	A: ● D: ●●●● T:	A: ● D: ●● T: ●●		
		DD DR LT 4	DD DR LT 4	DD DR LT 2	DD DR LT 1	DD DR LT 1		
				Web email verification		Login		
				A: ●● D: ●● T:		A: ● D: ●● T: ●		
				DD DR LT 2		DD DR LT 2		
				Business profile basics		Email verification - BE		
				A: ●● D: ● T:		A: ● D: ●●● T: ●		
				DD DR LT 3		DD DR LT 1		

Day 6

Backlog	Ready	Analyze		Develop		Test (3)	Ready to Deploy (4)	Deployed
		Doing (2)	Done	Doing (4)	Done			
Customer settings - Main Customer settings - My Addresses Customer settings - Booking History Customer Settings - Get Help & Legals Customer Settings - Notifications WEB CUSTOMER SETTINGS MOBILE BOOKING WEB BOOKING		Working hours A: ● D: ●● T: ● DD DR LT 6		Business profile basics A: ●● D: ●● T: ● DD DR LT 3	Forgot password A: ●● D: ●●● T: ● DD DR LT 2	Login A: ●● D: ●●● T: ●● DD DR LT 2	Registration A: ●● D: ●●● T: ●● DD DR LT 1	
		Payment basics A: ●● D: ●● T: ● DD DR LT 6		Add services and prices A: ●● D: ●● T: ● DD DR LT 4		Email verification - FE A: ●● D: ●●●● T: ● DD DR LT 1	Email verification - BE A: ●● D: ●●●● T: ● DD DR LT 1	
				Upload photos A: ●● D: ●● T: ● DD DR LT 4		Web email verification A: ●● D: ●●● T: ● DD DR LT 2		

Day 7

Backlog	Ready	Analyze		Develop		Test (3)	Ready to Deploy	Deployed
		Doing (2)	Done	Doing (4)	Done			
Customer settings - Booking History Customer Settings - Get Help & Legals Customer Settings - Notifications WEB CUSTOMER SETTINGS MOBILE BOOKING WEB BOOKING		Customer settings - Main A: ● D: ●●● T: ●● DD DR LT 7	Payment basics A: ●● D: ●●● T: ●● DD DR LT 6	Business profile basics A: ●● D: ●●● T: ●● DD DR LT 3 Upload photos A: ●● D: ●● T: ●● DD DR LT 4 Working hours A: ● D: ● T: ● DD DR LT 6	Add services and prices A: ●● D: ●● T: ●● DD DR LT 4	Email verification - FE A: ● D: ●●● T: ●● DD DR LT 1	Login A: ● D: ●● T: ●● DD DR LT 2	Registration A: ● D: ●● T: ●● DD DR LT 7 - 1 = 6
		Customer settings - My Addresses A: ● D: ●● T: ●● DD DR LT 7				Forgot password A: ●● D: ●●● T: ● DD DR LT 2	Web email verification A: ●● D: ●● T: ● DD DR LT 2	Email verification - BE A: ● D: ●●● T: ● DD DR LT 7 - 1 = 6

Day 8

Backlog	Ready	Analyze		Develop		Test (3)	Ready to Deploy (4)	Deployed
		Doing (2)	Done	Doing (4)	Done			
Customer Settings - Notifications Web customer settings - My Addresses Web customer settings - Password Web customer settings - Main Web customer Settings - Notifications MOBILE BOOKING WEB BOOKING		Customer settings - Booking History A: ● D: T: DD DR LT 8	Customer settings - My Addresses A: ●● D: T: DD DR LT 7	Upload photos A: ●● D: ●●● T: DD DR LT 4		Add services and prices A: ●● D: ●● T: ● DD DR LT 4	Email verification - FE A: ● D: ●●● T: ●● DD DR LT 1	Registration A: ● D: ●● T: ●● DD DR LT 7 - 1 = 6
		Customer settings - Get Help & Legals A: ● D: T: DD DR LT 8		Working hours A: ● D: ●● T: DD DR LT 6		Business profile basics A: ●● D: ●●● T: ● DD DR LT 3	Forgot password A: ●● D: ●●● T: ● DD DR LT 2	Email verification - BE A: ● D: ●●● T: ● DD DR LT 7 - 1 = 6
				Payment basics A: ●● D: ● T: DD DR LT 6				Login A: ● D: ●● T: ●● DD DR LT 8 - 2 = 6
				Customer settings - Main A: ● D: ● T: DD DR LT 7				Web email verification A: ●● D: ●● T: ● DD DR LT 8 - 2 = 6

Day 9

Backlog	Ready	Analyze		Develop		Test (3)	Ready to Deploy (4)	Deployed
		Doing (2)	Done	Doing (4)	Done			
Web customer settings - My Addresses Web customer settings - Password Web customer settings - Main Web customer Settings - Notifications MOBILE BOOKING WEB BOOKING	Web customer settings - My Addresses Web customer settings - Password	Customer settings - Booking History A: ●● D: T: DD DR LT 8	Customer settings - Get Help & Legals A: ●● D: T: DD DR LT 8	Working hours A: ● D: ●●● T: DD DR LT 6	Upload photos A: ●● D: ●●●● T: DD DR LT 4	Business profile basics A: ●● D: ●●● T: ●● DD DR LT 3	Email verification - FE A: ● D: ●●●● T: ●● DD DR LT 1	Registration A: ● D: ●● T: ●● DD DR LT 7 - 1 = 6
		Customer settings - Notifications A: ● D: T: DD DR LT 9		Payment basics A: ●● D: ●● T: DD DR LT 6	Customer settings - Main A: ● D: ●● T: DD DR LT 7		Forgot password A: ●● D: ●●● T: ● DD DR LT 2	Email verification - BE A: ● D: ●●● T: ● DD DR LT 7 - 1 = 6
				Customer settings - My Addresses A: ●● D: ● T: DD DR LT 7			Add services and prices A: ●● D: ●● T: ● DD DR LT 4	Login A: ● D: ●● T: ●● DD DR LT 8 - 2 = 6
								Web email verification A: ●● D: ●● T: ● DD DR LT 8 - 2 = 6

Day 10

Backlog	Ready	Analyze		Develop		Test (3)	Ready to Deploy (4)	Deployed	
		Doing (2)	Done	Doing (4)	Done				
	Web customer settings - Password	Web customer settings - My Addresses A: ● D: T: DD DR LT 10	Customer settings - Notifications A: ●● D: T: DD DR LT 9	Payment basics A: ●● D: ●●● T: DD DR LT 6 Customer settings - My Addresses A: ●● D: ●● T: DD DR LT 7 Customer settings - Get Help & Legals A: ●● D: ● T: DD DR LT 8 Customer settings - Booking History A: ●● D: ● T: DD DR LT 8	Working hours A: ● D: ●●●● T: DD DR LT 6	Upload photos A: ●● D: ●●●● T: ● DD DR LT 4 Customer settings - Main A: ● D: ●● T: ● DD DR LT 7	Business profile basics A: ●● D: ●●● T: ●● DD DR LT 3	Registration A: ●● D: ●● T: ●● DD DR LT 7 - 1 - 6 Email verification - BE A: ● D: ●●● T: ● DD DR LT 7 - 1 - 6 Login A: ● D: ●●● T: ● DD DR LT 8 - 2 - 6 Web email verification A: ●● D: ●● T: ● DD DR LT 8 - 2 - 6 Email verification - FE A: ● D: ●●●● T: ●● DD DR LT 10 - 1 - 9 Forgot password A: ●● D: ●●● T: ● DD DR LT 10 - 2 - 8 Add services and prices A: ●● D: ●● T: ● DD DR LT 10 - 4 - 6	
	Web customer settings - Main								
	Web customer Settings - Notifications								
Booking - Time slot									
Selecting and adding an address									
Working hours and service location(s)									
Booking - Payment									
Booking details - Summary									
WEB BOOKING									

Cumulative Flow Diagram

Day	Backlog	Ready	Analyse	Develop	Test	Ready to Deploy	Deployed
Day 1		9	3	3	0	0	0
Day 2		7	2	3	3	0	0
Day 3		6	2	3	4	0	0
Day 4		8	2	3	5	1	0
Day 5		7	3	2	4	3	0
Day 6		6	2	2	4	3	2
Day 7		4	2	3	4	2	2
Day 8		5	2	3	4	2	4
Day 9		4	2	3	5	1	3
Day 10		6	3	2	5	2	1

Day	Backlog	Ready	Analyse	Develop	Test	Ready to Deploy	Deployed
Day 0	15	0	0	0	0	0	0
Day 1	15	6	3	0	0	0	0
Day 2	15	8	6	3	0	0	0
Day 3	15	9	7	4	0	0	0
Day 4	19	11	9	6	1	0	0
Day 5	19	12	9	7	3	0	0
Day 6	19	13	11	9	5	2	0
Day 7	19	15	13	10	6	4	2
Day 8	22	17	15	12	8	6	4
Day 9	22	18	16	13	8	7	4
Day 10	26	20	17	15	10	8	7

