Project management tools in software development

Master of Science (Tech) Thesis
University of Turku
Department of Computing, Faculty of Technology
Information and Communication Technology Degree Program
2025
Tuomas Santa

Supervisors:
Tuomas Mäkilä (UTU)
Jari-Matti Mäkelä (UTU)

The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin Originality Check service.

UNIVERSITY OF TURKU
Department of Computing, Faculty of Technology

TUOMAS SANTA: Project management tools in software development

Master of Science (Tech) Thesis, 120 p.
Information and Communication Technology Degree Program
October 2025

Project management tools play a critical role in modern software development. They boost the efficiency of planning, executing and delivering software projects. Some tools support software development related methodologies such as Agile, Waterfall and DevOps by facilitating task tracking, team collaboration and workflow automation. However, the tools effectiveness depends on their alignment with project requirements, team dynamics and organizational processes. This thesis evaluates three widely used project management tools: GitLab, Jira and Microsoft Project. A survey was also conducted with 110 industry professionals to gain more information about tool use. This thesis examines the tools' strengths and limitations in a simulated software project and addresses key questions about their necessity, benefits and suitability for various project types. The findings reveal that there is no single optimal tool. The most suitable tool depends on factors such as methodology, team size, and integration needs. GitLab, Microsoft Project and Jira all offer different strengths and weaknesses. This thesis concludes that effective project management requires a contextualized approach, with tools selected based on suitability rather than perceived superiority. It also emphasizes the growing importance of automation, AI integration and hybrid tool ecosystems in boosting productivity. By offering practical guidance on tool selection and optimization, this thesis aims to help teams and organizations enhance their software development workflows.

Keywords: Project management, Project management tools, Software Development, Jira, Microsoft Project, GitLab

Contents

1 Introduction **1**

2 Project management **6**

2.1 Project management in general 7

2.2 Project management in software development 10

3 Software development methodologies **14**

3.1 Waterfall 16

3.2 Agile and Scrum 18

3.3 DevOps 24

3.4 SAFe and LeSS 27

3.5 Kanban 30

4 Project management tools in software development **33**

4.1 Gitlab 36

4.1.1 Project management methodologies support 39

4.1.2 Code collaboration 42

4.1.3 Scalability 44

4.1.4 Regular project management tasks 46

4.2 Jira 48

4.2.1 Project management methodologies support 52

4.2.2 Code collaboration 55
4.2.3 Scalability 59
4.2.4 Regular project management tasks 61
4.3 Microsoft project 65
4.3.1 Project management methodologies support 68
4.3.2 Code collaboration 72
4.3.3 Scalability 73
4.3.4 Regular project management tasks 76
4.4 Comparative summary of the tools 78
4.5 What AI tools and functions there are to help with project management? 82

5 Survey **84**
5.1 Study design 84
5.2 Results 86

6 Discussion **101**
6.1 Summary of key findings 101
6.2 Comparison with previous research 107
6.3 Answers to research questions 109
6.4 Future insights 113

7 Conclusion **118**

References **121**

A Appendices **128**
A.1 Survey screenshots 128

List of Figures

2.1 Project management principles based on [5] 8

3.1 Waterfall methodology based on [16] 16
3.2 Agile methodology based on [18] 19
3.3 Scrum framework based on [21] 21
3.4 DevOps lifecycle based on [30] 25
3.5 SAFe by framework.scaledagile.com [35] 28
3.6 Kanban board based on [39] 31

4.1 GitLab Epics [41] 37
4.2 GitLab Issue Board 38
4.3 GitLab milestones 39
4.4 GitLab CI/CD pipeline analytics 43
4.5 Jira backlog from a sample project 49
4.6 Jira sprint board 50
4.7 Jira timeline 51
4.8 Jira calendar 52
4.9 Jira dependencies 54
4.10 Jira and GitLab CI/CD integration 57
4.11 Jira integrated with Slack. The view from Slack app. 62
4.12 Microsoft Planner 67
4.13 Microsoft Teams Kanban-style board 70

4.14 Microsoft Project resource allocation 74
4.15 Microsoft Project Gantt-chart 76
4.16 Comparison of the tools based on previous findings 80

5.1 Survey participants showing the results of experience in software development and location. 86
5.2 Survey respondents' role in a company and size of their organisation . 87
5.3 Industry Sectors Represented by Participants 88
5.4 Project management methodologies used 89
5.5 Project management tools used 90
5.6 Primary project management tool in use 92
5.7 Most valued features in project management tools 94
5.8 Challenges encountered with current project management tools 95
5.9 Evaluation of tool-workflow compatibility and support (Scale: 1 = Poor, 5 = Excellent) 97
5.10 Interest in AI-powered features in project management tools 98

List of Tables

4.1 Jira CI/CD integration 58

1 Introduction

In today's technology driven world, software development plays a key role in shaping consumer and enterprise systems and products alike. Different software products have become the foundation of modern innovation and the standard of living. There is a demand for high-quality software products that can be constructed quickly. Effective management of these projects has become increasingly important.

Significant advancements have been made in development methodologies and tools. Nevertheless, software projects often struggle with late delivery, budget overruns, and reduced product quality. One of the main reasons for these challenges is ineffective project management. A variety of tools and frameworks exist to support the planning and execution of software projects. However, it is still unclear which tools best suit different project environments and how these tools influence the overall workflow.

Software project management refers to the application of processes, methodologies, knowledge, skills and tools to ensure the successful delivery of software products. Software development is dynamic and iterative in nature, therefore it requires flexibility and rapid adaptation. In this context, project management tools and methodologies such as Agile, Kanban and DevOps offer structured approaches to navigating projects from start to finish.

The aim of this thesis is to explore and evaluate different project management tools used in software development. The objectives are:

- To identify commonly used project management tools in software development.
- To assess the benefits and limitations of each tool in different project contexts.
- To analyze how specific tools impact team collaboration, workflow efficiency and project success.

The motivation for this thesis stems from several real-world observations. A notable trend in job requirements within the software industry is the wide variety of project management tools that professionals are expected to use. This raises important questions about the necessity and efficiency of using multiple tools at the same time. Specifically, it raises questions about whether using multiple tools leads to redundancy or inefficiencies in project workflows. Could a more streamlined approach be achieved? The potential for automating and integrating these tools through unified platforms could enhance workflow efficiency and reduce cognitive overhead. This thesis will explore these possibilities in the context of software project management.

This leads to the research questions in this thesis. Research questions were chosen to answer some of the concerns regarding project management and personal real-world observations. This thesis is guided by the following research questions:

- **RQ1:** What are the necessary tools in software project management?
- **RQ2:** What benefits does each tool bring to the team?
- **RQ3:** Is there a universal project management tool in software development?

This thesis takes a mixed-methods approach. It includes a survey to gather real-world insights from practitioners in the field. This is complemented by practical testing and analysis of selected tools via a simulated sample project. Three tools were chosen for this thesis: GitLab, Jira and Microsoft Project. The tools were chosen for the popularity and differences in terms of functions. The selection of

tools was also influenced by the availability of free trials and personal prior experience with Jira and GitLab. While GitHub and Azure DevOps would have been viable alternatives, GitLab was prioritized since it is provided by the University. Azure DevOps was not examined in detail beyond its role in the 'Microsoft Project' section, as including a comparison of four different tools seemed excessive in terms of the length of the thesis. The methodology used to evaluate the tools within the simulated to-do list application comprised three main components: (1) testing the application and making personal observations about the tools and their functionalities; (2) reviewing existing academic and industry literature to support and expand the findings; and (3) examining the official documentation provided by the respective companies.

The findings of the testing and survey will also be compared with those of other related surveys, articles and studies. The findings are evaluated in the context of different software project environments. The tools to be utilized in the simulated project were selected and made first. Based on these selections, the survey was then designed. This sequencing was intentional, as it allowed for the inclusion of survey questions that would directly complement and answer some of the questions in the simulated project.

AI-tools were used in the making of this thesis. DeepL was used to refine the grammar and find more appropriate synonyms. DeepL was particularly used in the early chapters. Other AI tools were also used as synonym generators or as tools that could find the correct wording for specific contextual needs. These were DeepL, QuillBot and pharaphaser. ChatGPT and integrated AI-based Google search tool were used to find relevant articles and books in addition to more conventional sites like Google Scholar, ResearchGate and IEEE Xplore. The grammatical tools in Overleaf and Word were also used to check for grammatical errors. ChatGPT and other AI tools and integrations were also used with the creation of the simulated

sample project.

The search process for the relevant articles and material began with IEEE Xplore, ResearchGate, and Google Scholar. Used keywords were “project management tools AND software development,” “agile project management,” "Project management AND software development," "Agile AND project management tools," and “DevOps project management”. In addition to that, some articles about specific tools were also searched for the later chapters: "Jira AND software development" and "Microsoft project AND agile", for example. Used articles were selected based on criteria such as publication date (preference for the last 10 years), relevance to software development practices, and whether they discussed concrete tools (e.g., Jira, GitLab, Microsoft Project, GitHub) rather than only theoretical frameworks. There were quite few articles that discussed and compared some of the tools. Other related surveys were also searched to contrast the results of this thesis with those of earlier surveys. Due to the shortage of earlier surveys, all the surveys found were used, regardless of their slight differences.

This thesis aims to improve understanding of the influence of project management tools on software project outcomes. This thesis also aims to provide practical guidance to software teams and organizations on selecting appropriate project management strategies by identifying the strengths and limitations of these tools in specific contexts.

The remainder of this thesis is structured as follows:

- **Chapter 2** reviews the theoretical background of project management in software development and in general.
- **Chapter 3** introduces key methodologies used in software development.
- **Chapter 4** creating a sample project with tool analysis.
- **Chapter 5** survey and its findings.

- **Chapter 6** discussion, answers to research questions and future insights.
- **Chapter 7** conclusion.

2 Project management

Project management is an essential part of any project. In this chapter, this thesis takes a closer look at its history and standard practices, with a specific focus on software development.

As a formal discipline, project management has evolved significantly over time. In the early 1900s, Frederick Taylor introduced scientific management principles emphasizing efficiency and standardized workflows.[1]. Taylor also emphasized dividing work into planning and execution[2].

Key milestones were reached in the 1950s and 1960s with the development of formal tools such as the Critical Path Method (CPM)[3] and the Program Evaluation and Review Technique (PERT)[4]. These innovations enabled project managers to plan, schedule and control large-scale projects more effectively.

In the late 20th century, project management expanded into various industries following the establishment of professional organizations such as the Project Management Institute (PMI) in 1969. The PMI introduced the Project Management Body of Knowledge (PMBOK).[5] More recently, methodologies such as Agile and DevOps have emerged. Project management continues to evolve with technological advancements and new frameworks that emphasize collaboration, flexibility and value delivery.

2.1 Project management in general

Project management is essential for successfully completing tasks within defined constraints, such as time, cost, and quality. According to the Project Management Institute (PMI), a project is defined as 'a temporary endeavor undertaken to create a unique product, service or result'[5]. This definition highlights two key features of projects: they are limited in duration and aim to produce a unique outcome. Unlike routine operations, projects are non-repetitive, characterized by specific objectives, timelines, stakeholder involvement, and designated leadership[5]. This section takes a closer look at project management in general. This section provides basic information that will be expanded upon in future chapters and sections.

Project-based methodologies are now commonplace in many fields. These include construction, software development, and academic research. Project management involves structuring work efforts, enhancing coordination, and increasing the likelihood of achieving the desired results.

At its core, project management involves integrating a combination of knowledge, skills, tools and techniques in order to effectively meet project requirements. According to the PMI, effective project management enhances predictability, improves scheduling and optimizes risk and constraint management. In addition to that, projects lacking structured management may suffer from poor quality, missed deadlines, budget overruns and reputational damage. [5]

Figure 2.1: Project management principles based on [5]

The approach to project management can vary significantly depending on the organization, the nature of the project and the individuals involved. The Project Management Body of Knowledge (PMBOK) framework outlines nine key knowledge areas considered essential for successful project execution: integration, scope, time, cost, quality, human resources, communication, risk, and procurement management[5]. This is visualized in figure 2.1.

Project integration management involves coordinating every aspect of a project. This includes resources, stakeholders and deliverables. Integration ensures cohesive operations and a rapid response to bottlenecks or disruptions in complex environments involving multiple teams or departments. [5]

According to Khan[6], managing the project scope is one of the most critical responsibilities of a project manager. The scope of a project defines its boundaries, specifying what is and isn't included. It is closely linked to the quality and allocation of resources. As scope may evolve due to constraints or emerging requirements, effective management is essential to maintain project alignment and avoid uncontrolled growth. This is often referred to as 'scope creep'.[6]

Projects have clearly defined start and end points. Therefore, effective time man-

agement is essential to ensure that deliverables are completed within the allocated timeframe. Changes to the project scope can directly impact the timeline, so effective time management is essential for meeting deadlines and managing dependencies between project components.

Quality and risk management are designed to establish and maintain standards for project outcomes. Quality management involves planning, assurance and control to ensure that the final deliverables meet predefined criteria. Risk management involves identifying, analyzing and responding to potential project risks. Together, quality and risk management help to reduce costs, enhance consistency, and improve overall project success according to PMI.[5]

The aim of human resource management and communication management is to create an optimized, collaborative working environment. Clear role definitions improve resource allocation and reduce inefficiencies. This allows team members to focus on tasks that align with their expertise. Effective communication facilitates the flow of information within the team and with external stakeholders. This should lead to smoother collaboration and faster decision-making. [7] This is important in software development.

Procurement management involves acquiring the external goods and services needed to execute a project. This can entail anything from hiring third-party vendors to procuring technical infrastructure.[5]

The methodologies used for project management can vary widely depending on organizational culture, industry standards and the specific needs of the project. Some approaches include Agile, Lean, Waterfall, the Critical Path Method (CPM) and Critical Chain Project Management (CCPM). Agile frameworks are frequently adopted for software development projects due to their iterative and adaptive nature.[7]

In summary, project management is vital for delivering successful products and

services. Its objectives include creating customer value, streamlining internal processes, reducing costs and eliminating inefficiencies. Effective project management enables teams to prioritize important tasks, adds value, and improves performance and stakeholder satisfaction.

2.2 Project management in software development

Software development projects have unique characteristics, challenges and management approaches that set them apart from traditional project types. Such projects require tailored methodologies that reflect the iterative, incremental and dynamic nature of the software development life cycle (SDLC). The SDLC usually involves stages such as gathering requirements, analysis, design, coding, testing, deployment and maintenance. Unlike traditional project management models, software development project management favors iterative methodologies that allow greater flexibility and responsiveness, such as Agile.[8] This section covers the fundamentals of project management in software development. It emphasizes the key differences between software projects and traditional project management approaches. The subsequent chapters will examine the specific tools and methodologies commonly employed in software development. These insights will lay the groundwork for subsequent discussions in the later chapters.

Execution of software project management differs substantially from traditional methodologies. In many software teams, there is often times no designated project manager. Instead, roles such as product owner, Scrum Master, Agile coach and development team members often share project responsibilities. This decentralization reflects the collaborative and adaptive ethos common to Agile practices. The Agile methodology will be explained in more detail in the next chapter. [9]

One of the most significant differences between traditional project management and software-specific project management lies in how requirements are managed. In

conventional projects, requirements are usually defined at the beginning and remain stable throughout. In contrast, the nature of software projects means they operate under frequent requirement changes due to shifting user needs, evolving technology, or iterative discovery during development. This fluidity or agility demands processes that support change rather than resist it. This is important in the software field, where improvements and technological advancements occur relatively rapidly. It is vital for organizations to keep up with this fast-paced environment. This is emphasized by Fernandez [10].

Software projects often use incremental and iterative development approaches. These approaches allow work to be completed in smaller, more manageable segments, with continuous stakeholder feedback. This style or ideology enhances customer involvement and increases the ability to respond to emerging requirements. In contrast, traditional projects tend to follow a sequential, phase-based approach. Therefore changes late in the process are more challenging and expensive to implement.[11] Consider, the construction of a bridge. The requirements for the bridge, such as its length, load capacity and location, are usually clear from the outset and rarely change throughout the project. In contrast, the requirements for software development projects often evolve. For instance, if the operating system updates from Android 6.4 to Android 8.10 during the development cycle, it may require changes. Adjustments will be needed to a program initially designed for Android 6.4. This dynamic nature makes flexibility essential in software project management.

Despite these differences, both software and traditional project management require effective communication and coordination. In software development, cross-functional teams comprising developers, designers, testers and analysts work closely together and typically operate under Agile principles rather than in a hierarchical management structure.[11] This can be complemented with managerial employees using more hierarchical ways of operating in the company. However, there are scaled

Agile frameworks and other scaled frameworks that show guidelines on how to operate for companies as well as teams.

The composition of software teams often differs from traditional project structures. Roles such as product owner, Scrum Master and Agile Coach are responsible for facilitating workflows instead of the project manager. However, the specific configuration of roles can vary depending on the organization and the chosen methodology. These will be explained in the next chapter.

A defining concept in modern software teams is self-organization. This principle has gained interest across industries, particularly in software development, where responsiveness and adaptability are important. In a self-organizing team members make decisions, solve problems, and adapt to changes independently without relying on centralized authority. This can replace or complement traditional roles such as project managers.[12] This approach enhances flexibility and agility by reducing the need for hierarchical approvals. It empowers team members to make decisions directly, enabling them to leverage their expertise fully. After all, a project manager does not always have more in-depth knowledge than the team members in every area. In theory, distributing decision-making utilities the team's collective knowledge more effectively. This could lead to better outcomes and more efficient problem solving. This applies not only to software development, but also to all domains involving specialized experts. Perhaps this could be a reason why Agile and other methodologies are used across domains.

The core of self-organization is autonomy. It enables individuals or teams to make decisions regarding their tasks and workflows. Autonomy fosters a sense of ownership and accountability among team members. According to Moe et al.[13], this may result in increased motivation, better collaboration, and higher-quality outcomes.

Moe et al.[13]identifies three types of autonomy: (1) External Autonomy: Decision-

making authority lies outside the team, typically with stakeholders or organizational leadership. (2) Internal Autonomy: The team collectively makes decisions about task allocation, planning, and execution. While responsibilities may still be assigned. The team retains decision-making power.(3) Individual Autonomy: Team members operate independently, often making decisions related to how they approach and complete tasks. This is particularly useful in remote or asynchronous work environments, where minimal interaction is needed.

Other important aspects of self-organizing teams include adaptability and the presence of a rapid feedback loop. As these teams handle much of their own planning and problem-solving, they can adapt relatively quickly to changes without waiting for managerial input. Feedback mechanisms such as retrospectives and continuous integration enable teams to identify issues and make iterative improvements.[12] This highlights the responsibilities that comes with autonomy.

In conclusion, self-organization empowers teams to use their collective intelligence and creativity. It gives flexibility, promotes faster delivery cycles and enhances collaboration. This makes it particularly well-suited for complex and fast-paced software development environments.

3 Software development methodologies

Software development methodologies provide structured frameworks to guide the planning, execution and delivery of software projects. They serve to manage complexity, enhance team collaboration and optimize resource utilization. This thesis focuses on selected methodologies, emphasizing their relation to project management tools. By examining their foundational principles, benefits and challenges, it aims to develop a comprehensive understanding of how these methodologies influence the choice of how project management tools are implemented in software development contexts.

The Waterfall methodology is a traditional and linear approach to project management. It was historically prevalent in environments where project requirements were well-defined and stable. It is still used in more conventional or highly regulated industries and perhaps used partially in managerial positions on all domains. The Waterfall approach proceeds through distinct, sequential phases, such as requirements gathering, design, implementation, testing, and maintenance with minimal overlap or iteration. Its predictability and structured nature make it suitable for projects with fixed scopes and clear deliverables.

By contrast, Agile methodologies such as Scrum have become the dominant approach to software development. Agile frameworks emphasize flexibility, collabo-

ration and iterative delivery. They encourage frequent stakeholder engagement and continuous feedback loops, as well as the ability to respond to evolving requirements. The increasing adoption of Agile is attributed to its ability to enhance customer satisfaction, deliver incremental value and foster cross-functional team dynamics[14].

DevOps is another significant evolution of Agile of some sorts. It integrates software development and other operations to streamline the deployment pipeline. This methodology emphasizes close collaboration between development and operations teams. It also advocates automation, continuous integration and continuous delivery. DevOps aims to reduce deployment times, improve software quality and enhance operational efficiency. This aligns well with the demands of rapid delivery and Agile responsiveness.

In recent years, scaling Agile has become important as well. The Scaled Agile Framework (SAFe) and LeSS (Large-Scale Scrum) have gained prominence in addressing the challenges of applying Agile methodologies at scale. SAFe provides structured guidance on implementing Agile across multiple teams and departments within large organizations. It facilitates alignment, coordination and collaboration at enterprise level while preserving the core values of Agile. SAFe is particularly relevant for organizations looking to implement Agile practices on a large scale without compromising governance and strategic consistency.[14] In the next section this thesis will take a closer look on said methodologies. Lean and Kanban are also significant methodologies. However, those are methodologies that are used significantly outside of software development. Nevertheless, its underlying principles resonate strongly within software development contexts as well as other domains. These could be adding value and removing waste.

3.1 Waterfall

The Waterfall methodology is a traditional, structured approach to project management, characterized by a linear and sequential process. In this process each phase must be completed before moving on to the next one. The methodology is based on a set of principles that emphasize comprehensive documentation and clearly defined project stages. A main point of the Waterfall approach is establishing explicit objectives and requirements at the initiation of the project. Before development the project team engages in detailed planning and gathers all necessary information from stakeholders and clients. The Waterfall Method aims to minimize ambiguity and ensure that all project stakeholders share the same understanding of the project's goals by defining a clear set of objectives and requirements.[15] However, this can also be a disadvantage, since some projects may be difficult to plan in advance.

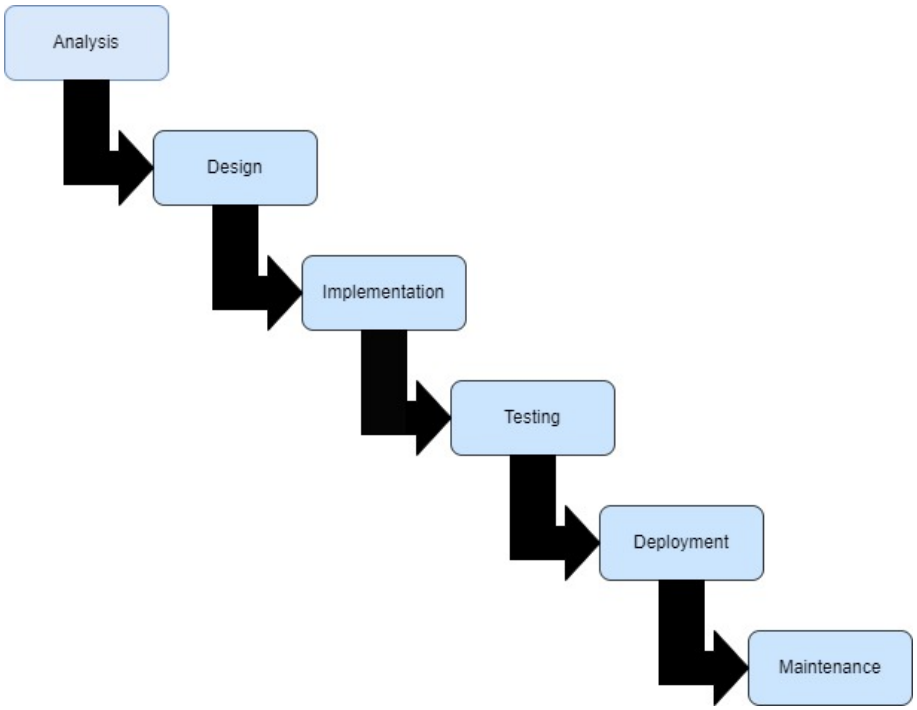


Figure 3.1: Waterfall methodology based on [16]

Figure 3.1 shows the basics of the waterfall method. The Waterfall method

involves sequential and phased project execution. Each phase must be completed before the next can begin, with limited opportunity for overlap or iteration between phases. The typical phases of the Waterfall Methodology include requirements analysis, design, implementation, testing, deployment and maintenance. This step-by-step approach provides a clear roadmap of progress which enables teams to plan resources and deadlines effectively. This benefits stakeholders too, since they can stay up to date with the project. The Waterfall Methodology is built around comprehensive documentation. Teams must develop documentation showing the project's progress, the decisions taken and the major deliverables during each phase. Examples of documentation include project plans, requirement specifications, design documents, test cases and user manuals. This is particularly useful for evolving teams, as they can access the necessary project information quickly.[16]

The Waterfall approach is predominantly non-iterative. This means that once a phase is completed, it is almost never revisited. Consequently, changes to requirements or design specifications introduced in later stages can result in significant cost implications and project timeline disruptions. While the rigid phase structure promotes stability and predictability, it can limit projects subject to evolving requirements. This is a common scenario in software development environments. The predictability of the Waterfall model is particularly valued in large-scale initiatives, where stakeholders demand clear assurances regarding budget adherence and timely delivery.

Another notable feature of the Waterfall methodology is the explicit specification of roles and responsibilities. Tasks within each phase are assigned to designated personnel. This supports deadline compliance and enhances managerial oversight. However, a significant limitation is the restricted involvement of end users or customers during the development process. User feedback is essential for producing high-quality, user-centric software. Without continuous engagement, the final prod-

uct may be misaligned with evolving user expectations or market needs. This issue is further compounded in dynamic development environments, where flexibility and responsiveness to change are paramount. [16]

In conclusion, the Waterfall methodology may serve as an effective project management approach in cases where project requirements are stable, well-defined and unlikely to change. However, its inherent rigidity and limited adaptability make it less suitable for contexts such where continuous iteration, user involvement and the capacity to respond to shifting requirements are important to project success. Some would argue that software development is this kind of context.

3.2 Agile and Scrum

The field of software development is evolving rapidly, driven by continuous technological advancements and the increasing demands of competitive markets. To address these dynamics and real-world issues software development methodologies must adapt to ensure efficiency, responsiveness, and quality. One of the most widely used modern approaches is the Agile methodology. It focuses on flexibility, collaboration and iterative improvement, and has therefore gained widespread adoption. Agile practices facilitate accelerated release cycles, which are essential in today's volatile market where hardware evolves rapidly and organizations strive to deliver innovations swiftly in response to competitive breakthroughs.[17] Figure 3.2 shows the basic fluidity of Agile methodology.

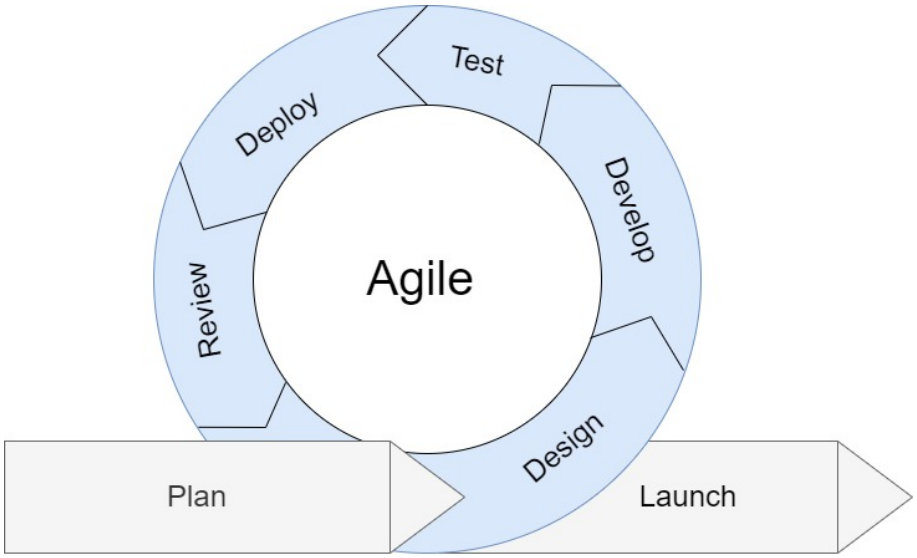


Figure 3.2: Agile methodology based on [18]

At the core of Agile is the Agile Manifesto, which outlines four fundamental values[19]:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a fixed plan

These values are supported by twelve guiding principles that emphasize speed, adaptability, simplicity and customer-centrality. These are promoted in the Agile manifesto[19]. Agile promotes environments in which cross-functional teams work collaboratively, continuously improve and prioritize the delivery of functional software [19].

Agile methodologies offer several distinct advantages. Agile teams can accommodate evolving user requirements and project adjustments without wasting significant resources. Short development iterations (sprints) and regular feedback cycles

help refine the product to better meet user needs [17]. Additionally, the Agile approach encourages frequent communication among team members and stakeholders. This includes daily stand-up meetings and regular progress reviews which are implemented in Scrum teams. These interactions help to synchronize development efforts, reduce duplication and enhance transparency.

Of the various Agile frameworks, Scrum is a structured yet adaptable approach to software development. It was developed in response to the lack of effective methodologies for managing complex software projects. A Scrum team is often referred to as a Agile Scrum team. It delineates three primary roles[20]:

- The Product Owner represents stakeholder interests and manages the product backlog
- The Scrum Master facilitates the Scrum process and ensures adherence to Agile principles
- The Development Team: a self-organizing group responsible for delivering incremental components of the product

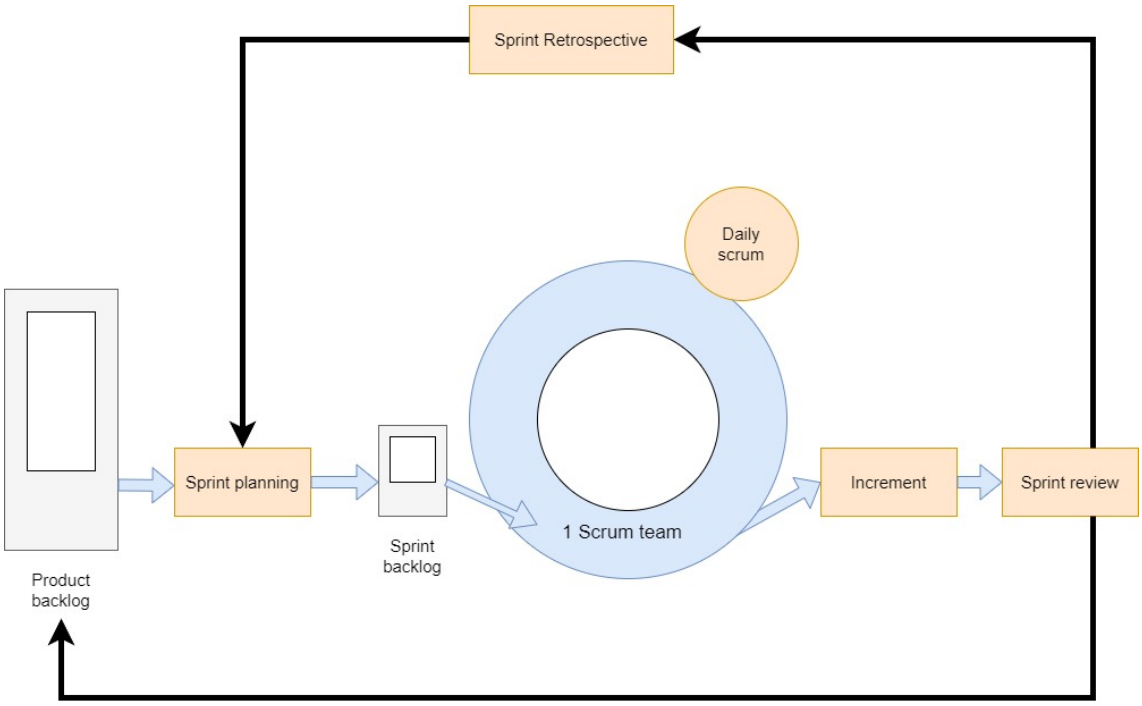


Figure 3.3: Scrum framework based on [21]

Scrum includes several artifacts central to the methodology as seen on Figure 3.3[20]:

- The Product Backlog is a dynamic, prioritized list of features, enhancements, and fixes required for the product.
- The Sprint Backlog includes selected items from the Product Backlog that are planned for implementation during a specific sprint.
- The Increment refers to the cumulative sum of all completed work that meets the definition of "done" and is potentially shippable or releasable at the end of a sprint.
- Daily scrum is a daily meeting between team members. This can also be transformed into weekly meetings.

- Sprint review and retrospective are essential on how to optimize processes and products in the future. These are made to optimize team dynamics, the product, workflows and working processes inside the team. Retrospective can include Agile coaches for example.

Scrum is characterized by strictly timed iterations (sprints), commonly ranging from two to four weeks[20]. Each sprint begins with Sprint Planning which is a collaborative meeting during which the development team and product owner determine the most valuable items to be addressed. During the sprint, Daily Scrum meetings (also called stand-ups) are held to coordinate tasks, report progress and address problems. These short meetings foster real-time synchronization and are designed to reduce duplicate work.[20] This value ideology can also be seen as a part of the Lean principles.

Once a sprint has been completed, a Sprint Review is held to evaluate the increment, gather feedback, and adjust the Product Backlog if necessary. This is followed by a Sprint Retrospective, which provides the development team with an opportunity to reflect on the sprint process, identify areas for improvement, and refine team practices. These feedback loops are essential for continuous improvement and adaptability, which are cornerstones of Agile style development.[20] This also keeps employees up-do-date on their colleagues and project status. Sprint Retrospective also can be used to discuss on specific tools that the team is using. Perhaps a team member would prefer another tool for a specific task. Team members can voice these concerns in the retrospective meeting.

Scrum is closely aligned with the principles of the Agile Manifesto[19], particularly its emphasis on customer collaboration and responsiveness to change. Its iterative structure enables teams to adjust swiftly to stakeholder feedback and maintain a strong focus on end-user value. Despite its strengths, Scrum is not without limitations. While it is highly effective in small to mid-sized teams, scaling Scrum

across large enterprises can present challenges. Furthermore, its successful implementation requires substantial change if the company is previously using more rigid management.

Agile and Scrum are inherently flexible methodologies that can be adapted to meet the specific needs of individual organizations. This adaptability may explain the diversity of interpretations and variations in their practical implementation and explanatory material and scientific research. For example Rigby et al.[22] takes a closer look on how agile has changed and how some people have different views and opinions about agile. Rigby et al.[22] presents a set of managerial perspectives that may at times be oversimplified or insufficiently substantiated. Even though they are intended to clarify Agile principles. In their effort to make Agile more accessible, they distill it into five guiding statements or guidelines: "Get everyone on the same page," "Don't change structures right away; change roles instead," "Name only one boss for each decision," "Focus on teams, not individuals," and "Lead with questions, not orders." These statements suggest that Agile should be understood less as a rigid methodology and more as an adaptable approach to organizing work, set of basic principles or an ideology. It should not be intended to constrain, but to enable and empower teams and give them more tools to improve their workflow.

The diversity in interpretation highlighted by Rigby et al.[22] also extends to broader academic and professional discourse on Agile. Additional confusion arises from the diverse variations in its implementations and readable material. This may reflect differences in temporal and contextual framing which creates variation on how people view agile. For instance, Kelly (2008) [23] presents a perspective on Agile that differs in nuance from more recent literature. Morcov et al. (2025) [24] for example, conceptualize Agile in a distinctly modern manner when compared to Kelly (2008) or even to the foundational principles articulated in the Agile Manifesto [25] [19]. This evolution in thought suggests that as the world changes, so too does

Agile. Software projects and the world have evolved significantly which may lead to some adjustments into Agile while the basic principles stay the same. Perhaps that is the beauty of continuous integration and improvement highlighted in Agile and in the overall domain.

Drawing on both real-world observations and prior literature, including Doz et al. [26], Solala [27], Bäcklund [28] and numerous other sources, it appears that Agile has extended well beyond its original association with IT-related fields. Evidence suggests that Agile principles are increasingly being applied in diverse domains, including public administration and governmental contexts in Finland. For instance, Bäcklund [28] discusses teams operating within non-IT settings in public administrations that adopt Agile practices, highlighting the methodology's adaptability and cross-sector relevance. This may also be a reason on why Agile principles need to be slightly adjusted, modified and perhaps generalized or simplified based on the target use case or sector. Agile manifesto closely targets software development as a target audience. Could it be generalized to all domains as an ideology that could guide teams and projects across domains? It could be useful in settings where managerial oversight is less needed such as in fields where specialized experts are the majority of the workforce. However, this raises questions. To what degree are practitioners, project managers and other decision makers able to recognize which parts or principles of a methodology are suitable for use? And which parts may be unnecessary or even harmful in a given context? Forcing Agile, lean or other methodologies may not always be the most optimal way of operating a company.

3.3 DevOps

DevOps has become a valid option to tackle the problems of project management. It is rooted in the words "development" and "operations". It is a methodology and organizational culture or set of principles that seeks to enhance collaboration,

increase automation and enable continuous delivery of software products [29]. DevOps advocates for a unified and integrated approach throughout the entire software lifecycle.

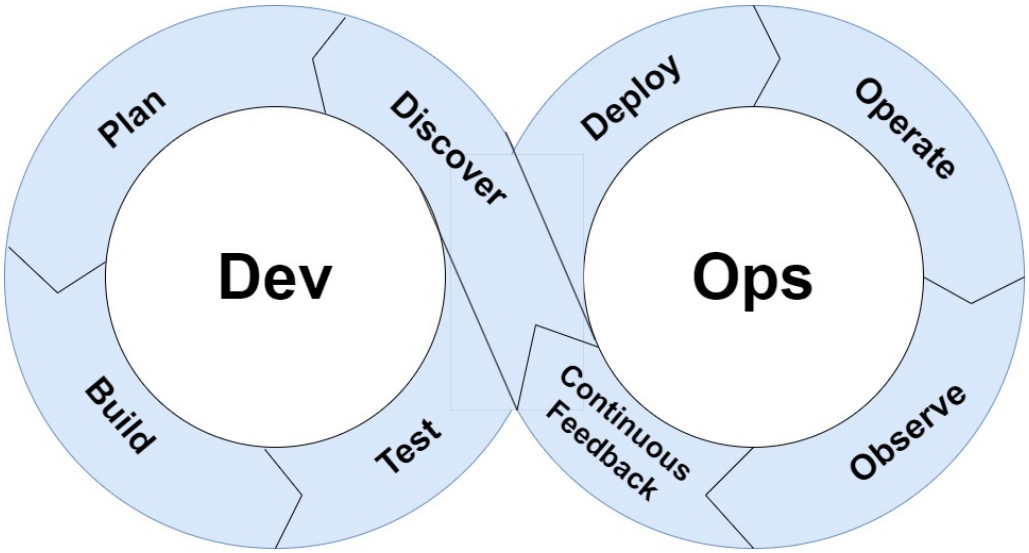


Figure 3.4: DevOps lifecycle based on [30]

At its core, DevOps aims to improve cooperation between developers, operations teams, and other stakeholders by promoting shared responsibility, better communication, and continuous feedback. This integrative mindset enables faster release cycles, greater system reliability and more efficient incident management[30]. DevOps fosters a culture of collaborative ownership, where development, testing, deployment, monitoring, and maintenance are shared and flexible responsibilities.

A typical DevOps team comprises professionals from software engineering and IT operations who work together throughout the product lifecycle. In many cases these roles overlap, forming cross-functional teams that can write code, configure infrastructure, run automated tests, and manage deployment pipelines. Responsibilities overlap and automation tools facilitate smoother transitions between phases. This leads to the distinction between “dev” and “ops” coming more and more blurred.

One of the defining characteristics of DevOps is its continuous and iterative

process, which is often visualized as an infinity loop (Figure 3.4). This cycle includes planning, development / building, testing, release and deployment, operation and monitoring, observing, continuous feedback and discovering problems. Each phase is closely linked to feedback mechanisms that enable teams to respond quickly to issues and ensure that software functions as intended in production environments. This loop reinforces a continuous improvement mindset, aligning with Agile principles and emphasizing rapid delivery without compromising stability.[17]

DevOps involves the adoption of various tools and practices, including[30]:

- Version control systems (e.g., Git: GitLab, GitHub)
- Continuous Integration/Continuous Delivery (CI/CD) pipelines
- Infrastructure as Code (IaC) tools (e.g., Terraform, Ansible)
- Automated testing frameworks
- Monitoring and observability platforms (e.g., Prometheus, Grafana)

These tools automate previously manual processes, reducing the risk of human error and supporting scalable deployment strategies. Automation also helps to maintain consistency between development and production environments. Chapter 4 of this thesis takes a closer look at these tools and their functions. Moreover, the DevOps approach enhances project visibility and control. Through practices such as real-time monitoring and automated alerts, teams can proactively identify system anomalies and performance bottlenecks [31].

Despite its many advantages, adopting DevOps requires significant organizational change. It often challenges existing workflows, hierarchical structures, and cultural mindsets. Successful implementation depends not only on technical tools, but also on fostering a culture of accountability, continuous improvement, and continuous learning. Active leadership support and commitment are required across

the organization. This may be one of the key points of creating a good working environment that produces great products with little to none wasted resources. These points are emphasized by Ebert et al.[30] This style of operating adds autonomy and highlights individual responsibilities across all team members and employers.

Similar to Agile, DevOps cannot be regarded as a straightforward or uniformly applied methodology. Literature reviews by Erich et al. [32] and Jabbari et al. [33] demonstrate that DevOps is implemented in different ways across projects, reflecting variations in organizational context, tools, and team practices.

In conclusion, DevOps is a methodology to the demands of modern software development and the notion of continuous improvement. It extends Agile principles beyond the development team to include operations. It also complements Agile methodologies, enabling end-to-end agility and responsiveness. DevOps methodology tries to ensure a workflow where software systems are built efficiently, deployed reliably, and maintained effectively over time.

3.4 SAFe and LeSS

Agile methodologies have become the predominant approach in software development. Scrum is one of the most widely adopted frameworks within this paradigm. Scrum's popularity stems from its emphasis on adaptability, iterative development, and self-organizing teams. However, as organizations grow in size and complexity, the limitations of traditional Agile frameworks become more apparent. Scaling Agile practices across multiple teams while maintaining alignment with broader organizational goals is a significant challenge. Against this problem, the Scaled Agile Framework (SAFe) gives a toolset. It extends Agile principles to large enterprises. [34]

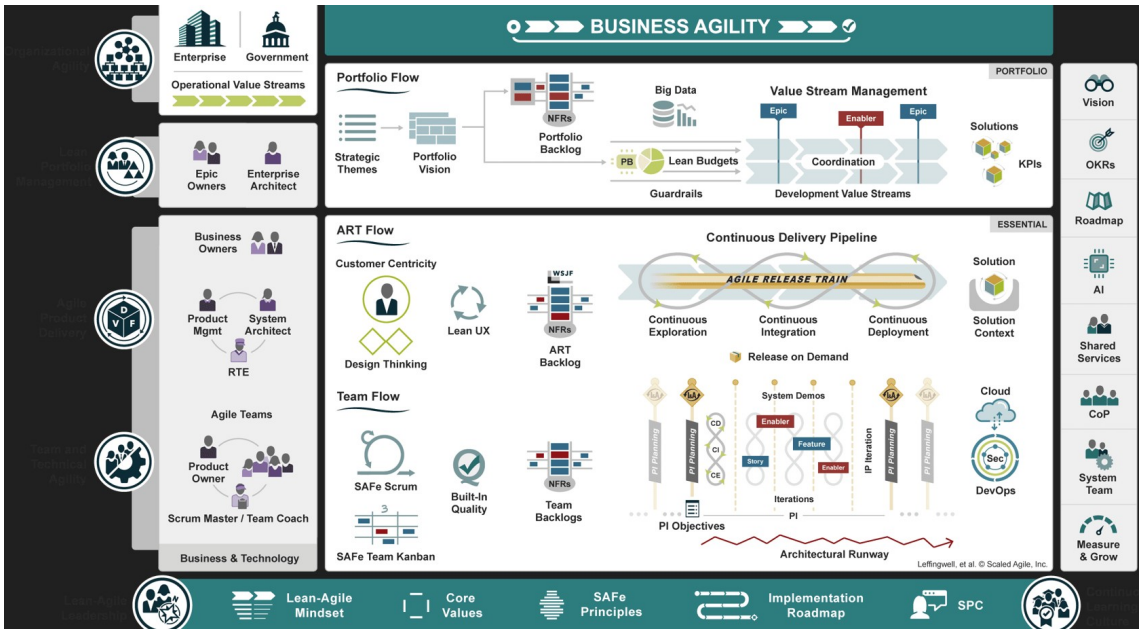


Figure 3.5: SAFe by framework.scaledagile.com [35]

Although Scrum is highly effective at team level, it does not inherently address collaboration or strategic alignment between different teams and departments. SAFe seeks to overcome these limitations by providing a framework that allows large organizations to implement Agile on a large scale. It introduces additional roles, structures, and coordination mechanisms that facilitate alignment across multiple teams and departments, and even organizational layers. These are visualized in the Figure 3.5. While maintaining the core values and principles of Agile and the Agile Manifesto, SAFe tries to implement those into the world of organizations.[19]

In Scrum, key roles include the Product Owner, Scrum Master and the Development Team. SAFe builds upon this foundation by introducing a broader hierarchy of roles such as[36]:

- Release Train Engineer (RTE): analogous to a chief Scrum Master, responsible for facilitating Agile Release Trains.

- System Architect/Engineer: oversees technical architecture and system-level decisions.
- Product Management: represents stakeholders and business needs across teams within the release train.

These roles enable scrum teams to be scaled up. The Agile Release Train (ART) is the operational core of SAFe. It is a cross-functional team of Agile teams that lasts for a long time. Scrum teams work around sprint cycles. However, teams often solve smaller problems during each sprint that collectively gain progress to the end product. As software projects may be larger in size, there are multiple teams that may operate on the same end-products. According to Putta et al.[34], ART often consists of 5 to 12 Scrum teams. It works collaboratively towards a shared project, delivering value in a synchronized manner. The ART provides a unified structure for planning, development, integration, and delivery. This endorses and promotes dependency management, improving transparency and fostering a shared understanding of objectives and deliverables among all participants.[34]

Moreover, SAFe promotes self-organization and empowerment within teams while providing strategic guidance and oversight at programming and portfolio levels. This approach enables organizations to be both flexible and disciplined. It enables them to adapt rapidly to change while remaining aligned with long-term goals.

LeSS (Large-Scale Scrum) is another scaled version of agile, similar to SAFe. It is a framework for scaling Scrum to multiple teams working on the same product. It builds on the core principles of Scrum, adapting them for use in larger, multi-team environments. Below is a brief overview of LeSS based on Almeida[37]: Foundation is Based on Scrum, keeping its simplicity and core roles (Product Owner, Scrum Master, Development Team). Scaling is solved with one Product Owner for all teams, one shared product Backlog and multiple Scrum teams. Coordination is solved via teams that synchronize their work through joint events like Sprint Planning, Sprint

Review, and Retrospective which is similar to agile. LeSS also emphasizes systems thinking, empirical process control, and continuous improvement.[37]

LeSS promotes decentralized decision-making, cross-team collaboration and a focus on customer value[37]. This makes it ideal for organizations looking to implement Agile at scale without introducing unnecessary complexity. The next chapters discuss LeSS and SAFe in more detail when implemented with tools.

In summary, while Scrum and Agile methodologies offer flexibility and iterative development, they are limited in large, complex environments without additional coordination mechanisms. SAFe and LeSS addresses this issue by providing a structured, scalable frameworks that retains the advantages of Agile while promoting organizational adaptability and efficiency[36].

3.5 Kanban

Kanban was originally developed in the 1950s by Toyota in Japan as a method to optimize inventory management in manufacturing processes. The core principle was to produce goods only when inventory levels required replenishment. This system was facilitated through the use of physical Kanban cards. These consisted of essential information such as what to produce, the quantity required and the destination of the products. [38] This method enabled a more efficient and demand-driven production workflow.

Since then, Kanban has evolved into a widely adopted framework within software development and project management across domains. It offers a set of guiding principles aimed at optimizing workflow efficiency. A Kanban board is shown in Figure 3.6. However, it can be modified according to the needs of teams and users. Kanban and Lean follow the same basic princibles as Kanban enabled the utilization of Lean. Both originate from the Toyota factories in Japan.

One of the foundational principles of Kanban is the visualization of the workflow.

This is typically achieved through the use of a Kanban board. The board consists of columns representing different stages of work and cards representing individual tasks or work items.[38] The visual structure enhances transparency and allows team members and stakeholders to understand the progress and status of tasks at a short glance. It helps not only in improving awareness but also in revealing dependencies, bottlenecks and opportunities for process enhancement[38].

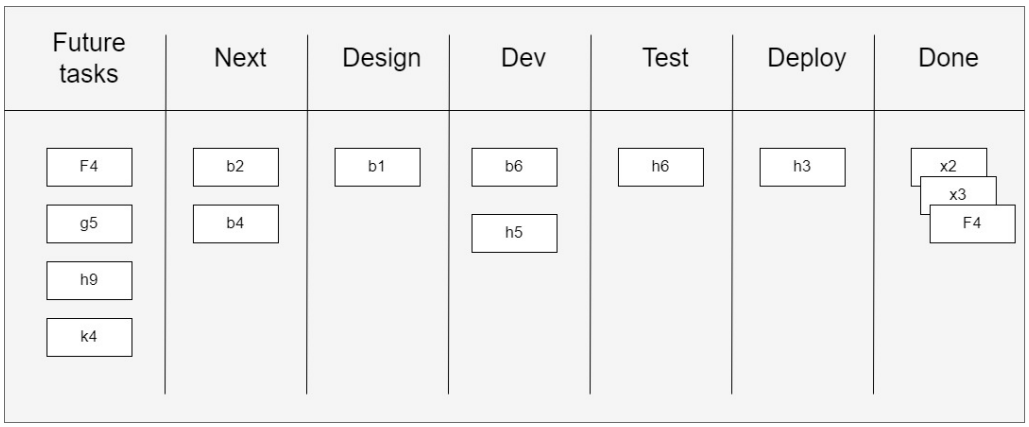


Figure 3.6: Kanban board based on [39]

A second key principle is limiting work in progress (WIP). Kanban restricts the number of tasks in each phase of the workflow. This reduces inefficiencies and prevent overburdening team members of work. [38] In software development, this means focusing on completing high-priority tasks in sequence rather than multitasking, which often leads to delays and quality issues. Limiting WIP helps to minimize context-switching and ensures that resources are allocated to work that directly contributes to the project's goals. This gives more control to the management team of the overall project and processes.

Managing and measuring flow is another core point of Kanban. Monitoring the movement of tasks across the workflow allows teams to identify and address bottlenecks, delays or uneven workloads. [38] In the context of software development, this insight is particularly important when certain teams or developers fall behind

schedule. Since it could put the overall project timeline and deliverables at risk. This can also be seen on many project management tools. Many of their features are based on Kanban boards even though use cases may vary quite significantly.

Kanban also emphasizes the importance of making process policies explicit and transparent. Clearly documented policies help team members to understand operational procedures and decision-making criteria. This transparency promotes alignment, accountability and the ability to adapt to changes in processes. When comparing Kanban, Agile and Scrum, some notable similarities can be seen. Kanban can be seen as a basis or a tool for Agile or Scrum. Scrum backlog and the overall workflow can be seen as part of a Kanban board. Kanban follows similar fluidity which can be seen on Agile and Scrum.

In conclusion, collaborative and continuous improvement is a central principle of the Kanban methodology, as Ahmad[38] explains. Kanban promotes an iterative approach to optimizing workflows, encouraging teams to refine their processes based on data and observation. The system also includes mechanisms for identifying inefficiencies. Therefore it fosters a culture within organizations that supports learning, adaptability and sustained performance improvement.

4 Project management tools in software development

In recent years, optimization has become a key focus for software development organizations. Organizations are under increasing pressure to optimize workflows, end products, operational styles, revenue streams and team collaboration and productivity. They employ a variety of methodologies, principles and tools to solve these problems. These tools play an important role across domains. While some projects benefit from structured management, software projects often require specific functionalities that distinguish them from other fields. As previously mentioned, software development teams can differ significantly in the methodologies they adopt. This further complicates tool selection. The primary objective of project management tools is to minimize inefficiencies, particularly those resulting from wasted time and miscommunication. Comprehensive platforms that consolidate multiple project management functions into a single and integrated system are becoming increasingly popular.

Project management tools are digital platforms designed to facilitate the planning, execution, monitoring and evaluation of projects[40]. These tools automate essential functions, promote collaboration and improve visibility of tasks and timelines. These tools have become indispensable for managing complex projects, offering real-time oversight, performance metrics, and resource allocation capabilities. Project

management has evolved from manual methods such as physical Gantt charts and spreadsheet-based planning to sophisticated software solutions offering a wide range of features, third-party integrations and customizable workflows.

This chapter focuses specifically on the functionalities required in project management tools for software development teams. It is important to note that even within a single organization team structures and workflows can vary widely. Some teams use traditional Waterfall models, while others adopt Agile, DevOps, or hybrid methodologies. Consequently, the tools they use must accommodate a variety of operational preferences and project requirements. This raises a key question: Is there a universally superior project management tool, or must tool selection be context-dependent?

Task management is a fundamental capability of any project management tool, whether used for software development or general project coordination. However, task management can be implemented in various ways. For example, Kanban boards provide a visual representation of ongoing work, while Gantt charts facilitate time-based planning and dependency tracking. In some cases, even simple shared calendars can function effectively for lightweight task coordination as shown in chapter 5.

In software development, certain features are indispensable. Code collaboration, version control and bug tracking are all essential components of the software development lifecycle. These features ensure quality assurance and facilitate iterative improvements, both of which are critical to delivering reliable and maintainable software products. Additionally, tools that support documentation, internal communication and project transparency are necessary.

This chapter provides a practical assessment of three widely adopted project management tools: Jira, Microsoft Project, and GitLab. These tools were selected due to their popularity, differences and alignment with various project management

methodologies. Jira is commonly associated with Agile methodologies and excels in issue tracking and sprint planning. Microsoft Project supports structured, plan-driven projects. GitLab is well suited to DevOps integrations.

The evaluation has two primary objectives: first, to assess the practical usability and features of each tool through direct experimentation; and second, to evaluate how well each tool supports common project management methodologies in the context of software development. Key areas of focus include planning, task tracking, workflow management, code collaboration, scalability, alignment with methodologies and cross-functional collaboration. The evaluation was based on observations made during the sample project, on application materials provided by the companies, and on existing research that compared the functionalities of the tools.

To support this evaluation, a small-scale sample project was developed to simulate real-world software development processes. This involved creating a simple, web-based To-Do List application with features such as user registration and authentication, task creation and editing, deadline reminders and the ability to export data. This small coding sample project provided a controlled environment in which to explore how each of the following tools handle the practical requirements of project planning and execution: How Jira, Microsoft Project and GitLab handle the practical requirements of project planning and execution? As the tools are all different, their approaches will vary depending on which tool is used. The sample project was designed with five team members. However, it was hypothetically expanded in certain sections where scalability was a critical factor, such as when assessing whether specific features would function effectively within a framework like SAFe. The findings related to scalability were accompanied with insights from existing research, which served to validate the hypothetical results and enhance their overall value. The sample project was initially created using a base coding project developed in GitLab and was subsequently expanded to other platforms primarily during 2024 to

support the needs of this thesis. The Microsoft Project component of the thesis was made in 2024 using a free trial version, although certain screenshots, particularly those of Planner and related tools, were captured in 2025. Similarly, the Jira section was developed in 2024, with some additional screenshots taken in 2025.

This sample project was created using various AI tools and features. ChatGPT, Microsoft Co-Pilot and Deepseek were employed for coding and project ideas. Some AI tools and features in Jira were also employed to create epics, project structures and timelines. The following sections will examine the strengths and limitations of each tool in detail, along with how they align with specific software development methodologies and organizational contexts.

4.1 Gitlab

GitLab was used extensively during the empirical phase of this thesis. It didn't just serve as a version control system, but also as a platform for managing the development of the To-Do List application. It provides an integrated DevOps environment that combines source code management, issue tracking, continuous integration/delivery (CI/CD) and project planning. This section outlines how GitLab was used to organize and manage the development process, from initial planning to delivery. While GitLab is undoubtedly a DevOps tool, could it also be used in environments where DevOps isn't as prevalent? GitLab's code collaboration features were also utilized in Microsoft Project and Jira for obvious reasons. A basic project created in GitLab can also be implemented and used in other applications. However, this only applies to certain features and tools.

The project began with the creation of a dedicated GitLab repository, which served as the central workspace for all source code, documentation and planning information. GitLab's integrated issue tracker was then used to manage development tasks. Each feature planned for the To-Do List application (e.g. task creation,

deletion of tasks, creating a UI) was defined as a separate issue.

A project board was set up using GitLab's built-in Kanban-style template. Columns such as 'To Do', 'In Progress', 'Review' and 'Done' represented the current state of each task. To structure development over time, GitLab milestones were defined to group related issues into phases. These milestones are similar to agile sprints and can be used as such. For instance, a milestone labeled 'sprint1' enables the team to manage and measure progress against particular development objectives within the agile methodology. Issues and epics should be assigned to these milestones. GitLab provides automatic progress indicators based on the number of completed versus remaining tasks.

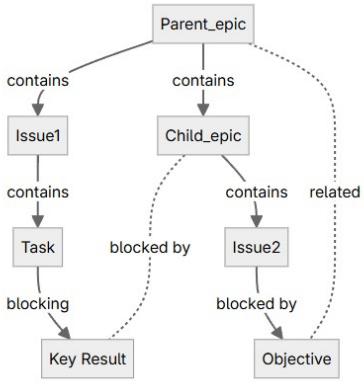


Figure 4.1: GitLab Epics [41]

Although creating epics is a premium feature in GitLab, the sample project was created using the student version. This may be because epics are usually used for larger projects and by larger companies. However, issues and tasks are still available in the student version. Epics work in a similar way to issues in that they can be seen as a parent to a child (see Figure 4.1). The next section provides a step-by-step guide on how the sample project was created.

1. **Creating a new GitLab project.**

   - `GitLab → New Project → Create Blank Project.`

- Project name (`todo-app-sample`)

2. **Defining a basic project structure.**
   - Creating repository folders, e.g., `/frontend`, `/backend`, `/docs`.
   - `README.md` file with a brief project description.
3. **Setting up initial labels and milestones.**
   - Labels such as `bug`, `feature`, and `documentation`.
   - Milestones (e.g., `Sprint 1`, `Sprint 2`).

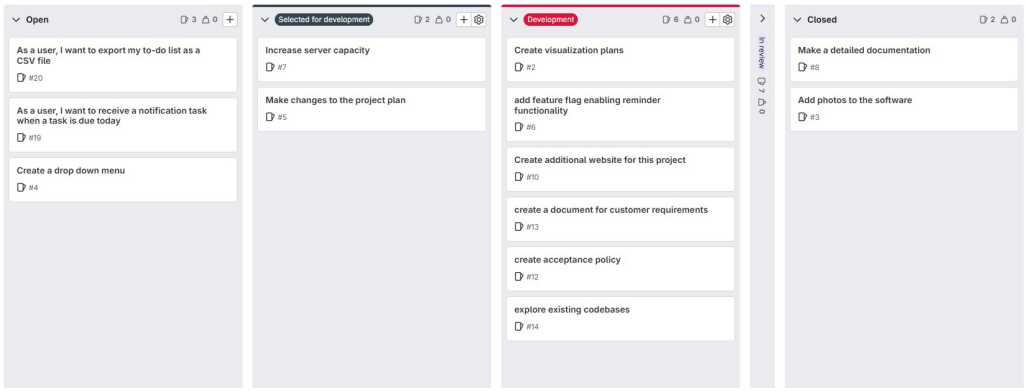


Figure 4.2: GitLab Issue Board

Each issue was assigned labels such as 'Frontend', 'Feature', 'Bug' or 'Documentation', which helped categorize work and filter views. GitLab enables role-based collaboration by allowing team members to be assigned to specific issues and merge requests. This feature was also employed in the sample project (see Figure 4.2). When developing specific features, issues were assigned to hypothetical team members. The team members then created a corresponding branch and linked it back to the issue. Discussions about tasks could be held directly in the comments section of

each issue or merge request. GitLab offers native integration of continuous integration and continuous deployment (CI/CD) pipelines. A '.gitlab-ci.yml' configuration file was created to support these needs.

GitLab also has many analytical tools integrated into the GitLab UI. These include CI/CD, code review, issue, team and progress tools. Some of these were briefly tested for the purposes of this thesis.

4.1.1 Project management methodologies support

GitLab is primarily known as an integrated DevOps platform, but it can accommodate various software project management methodologies. This section uses the sample project to evaluate the extent to which GitLab aligns with different methodologies, including Agile, DevOps, Waterfall, SAFe and Kanban. It analyses GitLab's capabilities and limitations in supporting these methodologies through its native features and tools.

Agile project management emphasizes iterative development, customer feedback and flexibility. GitLab is well-suited to Agile methodologies because of its features such as issue tracking, epics, milestones and boards. These provide the structural components necessary for organizing sprints and backlogs.

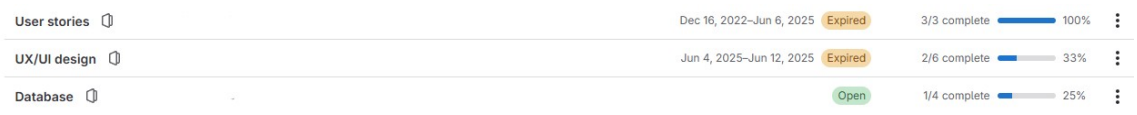


Figure 4.3: GitLab milestones

In the 'To-Do List application' project, tasks were created as GitLab issues and organized into milestones, which served as sprints (see Figure 4.3). Each milestone included a set of issues reflecting the planned tasks for that iteration. GitLab's boards were used to manage sprint planning and visualize the state of tasks across

columns, thereby replicating the flow of Agile ceremonies. GitLab is designed around the DevOps lifecycle, integrating source control, CI/CD, issue tracking, and deployment into a unified platform. It offers support for DevOps through features such as:

- Git repository management for version control
- `.gitlab-ci.yml` for defining release and deployment pipelines
- Auto DevOps, which automatically configures pipelines
- Management for testing and deployment and analytics tools

In the sample project, code changes were automatically pushed to branches and linked to issues, thereby triggering CI pipelines for testing. Deployments to test environments were automated using GitLab CI. This integration transformed GitLab from a planning tool into a delivery pipeline manager, aligning it closely with DevOps principles. However, there may be more suitable CI/CD solutions available outside of GitLab. Ailasmaa [42] documents a case in which a Finnish company initially employed two separate legacy tools for continuous integration (CI) and continuous delivery (CD). Following a transformation process that Ailasmaa closely observed, the company consolidated these tools by adopting Azure DevOps as an integrated CI/CD solution. However, these do not come without limitations. As Ailasmaa shows in his findings, every tool and tool combination has its advantages and disadvantages. However, in a simulated project setting created by Nguyen et al. [43], GitLab was found to be moderately effective in supporting CI/CD. However, the study did not include a comparison with alternative tools which limits the generalizability of its findings.

While GitLab is not inherently designed for the Waterfall method, it can be adapted to a certain extent to support linear and sequential project structures. The

milestone and dependency features allow sequential phases to be created, with each milestone having to be completed before the next one begins. In the sample project, a simplified waterfall model was simulated by dividing the project into distinct phases (e.g. planning, development, testing and deployment), with each phase represented as a milestone. Dependencies between issues were used to model the linear progression of work. GitLab's Gantt-style roadmaps (available in premium tiers) could be used to visualize this planning model. However, the sample project was not created using premium GitLab, so it was not tested in this thesis. Furthermore, GitLab lacks certain traditional waterfall tools, such as critical path analysis and baseline variance tracking. This makes it less suitable for teams that operate with Waterfall methodology.

SAFe is a scaled approach to scaling Agile across large enterprises and companies. It requires tools that support portfolio management, program-level coordination, and team-level execution. GitLab has features that align with SAFe, such as[44]:

- Epics and sub-epics, for program-level and portfolio-level planning
- Milestones and issues, for sprint and team-level planning
- Group-level visibility, allowing teams to coordinate work across projects

Epics could be used to model SAFe-style hierarchies, with major features broken down into smaller work items (issues) and assigned to different milestones. GitLab's group-level boards and dashboards support multiple teams. However, GitLab lacks value stream mapping and SAFe-specific reporting, both of which are often essential in SAFe environments. Integration with tools such as Jira Align or third-party portfolio management platforms may be necessary to enable broader use of SAFe.

Kanban emphasizes continuous flow, visual task tracking and limits to work in progress (WIP). GitLab natively supports Kanban through its issue boards. In this sample project, the Kanban board was used as the default project management

interface. Issues were manually moved between columns reflecting different workflow stages, and custom columns could be added to reflect additional states, such as 'Code Review' or 'Blocked'. Labels were used to filter views and apply custom WIP controls. The real-time updates, drag-and-drop UI and issue integration of GitLab make it a usable Kanban board. For instance, Kanban boards appeared to integrate smoothly with timelines.

In conclusion, GitLab provides support for Agile, DevOps, and Kanban methodologies. Mainly due to its integrated toolchain, flexible issue tracking, and automation features. It can reasonably simulate Waterfall workflows and offers basic alignment with SAFe. In the context of this test, GitLab proved most effective when used with Agile and DevOps.

4.1.2 Code collaboration

The effectiveness of GitLab in supporting code collaboration throughout the software development lifecycle was analyzed. It offers native, integrated features for version control, code review and CI/CD pipelines. This makes it a suitable platform for empirical testing, eliminating the need for third-party tools.

At the core of GitLab lies its built-in Git repository management system. Developers can create branches, manage merge requests (the equivalent of pull requests) and review code in the same environment used for task planning and tracking. This integration minimizes context switching and reduces the need for external coordination tools.

During the development of the sample project, the GitLab platform was used as the central hub for all code-related activities. The application's Git repository was hosted on GitLab, and collaborative development was structured around feature branches and merge requests (MRs). Each new feature or bug fix for the To-Do List application was developed in a dedicated branch named according to a con-

sistent convention (e.g. 'feature/add-task-form') and submitted for review via a merge request. This enables structured collaboration between team members, with all changes being reviewed, discussed and approved before merging into the main branch.

The GitLab merge request interface was used extensively throughout this project. Inline commenting allowed potential reviewers to provide specific feedback on code segments. Several potential rounds of simulated peer review took place during the development via comment features as a test for that feature.

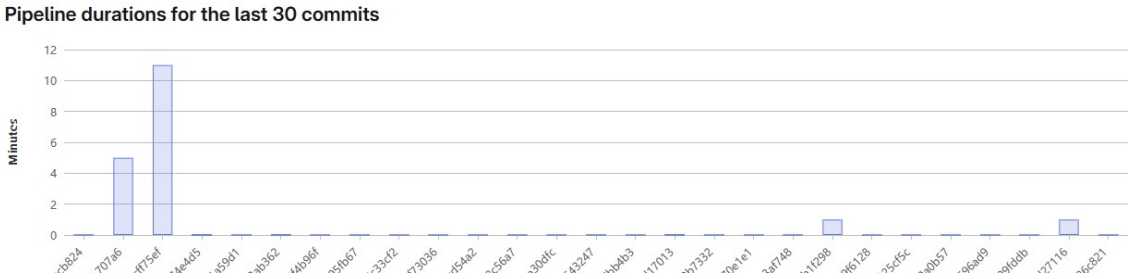


Figure 4.4: GitLab CI/CD pipeline analytics

Code collaboration was further supported by GitLab's native CI/CD pipeline, which was configured to perform automated tests and code quality checks on each merge request. This was defined in the '.gitlab-ci.yml' file and included stages for unit testing and build verification. These pipelines were automatically triggered on each commit and merge request (MR). This is particularly helpful during rapid iteration cycles, such as when implementing task prioritization and deadline tracking features in the application. GitLab also has analytic tools for various purposes, such as the CI/CD pipeline (Figure 4.4).

GitLab's issue tracking system was used to link specific code changes to tasks and requirements. Issues were created for each functional unit (e.g. 'Create Task List UI'), with commits linked to these issues via GitLab's shorthand notation.

This ensured that the implemented code could be traced back to its associated requirement, enabling potential reviewers and project stakeholders to understand the context of each change.

To facilitate visual and functional verification, the 'Review Apps' feature in GitLab could be configured to deploy the To-Do List application to a temporary testing environment for each merge request. Branching strategies play a significant role in collaborative workflows. This sample project followed a simplified GitFlow model, maintaining long main and development branches, as well as short feature and fix branches.

Overall, using GitLab in the development of the To-Do List application showed why Git-based platforms are useful when creating software. By centralizing version control, code review, CI/CD and issue tracking, GitLab enables developers to work cohesively without the friction of having to switch between tools. This centralized workflow of coding and project management activities could be particularly beneficial for small Agile and DevOps teams where fast iteration and clear communication are critical for delivering features within short timeframes. It is also essential tool in many code collaboration settings even when multiple other managerial tools are being used simultaneously.

4.1.3 Scalability

As organizations transition from small, single-team projects to cross-functional programs involving multiple agile teams, there is a growing demand for scalable practices such as LeSS (Large-Scale Scrum) and SAFe (Scaled Agile Framework), as well as for multi-team projects. This section evaluates the scalability of GitLab based on empirical observations made during the development of the To-Do List application, which simulated small and scaled team workflows. However, as the sample project was small and simulated, these findings are hypothetical.

GitLab's support for groups and subgroups facilitates the adoption of scalable team structures. Multiple hypothetical teams could be created within the same GitLab instance using separate subgroups, each with access to their own repositories, CI/CD pipelines and issue boards. These subgroups could be managed under a top-level group, which acts as the organizational entity.

This hierarchy enables ownership, permission control and resource partitioning, while still allowing visibility across the organization or multiple teams. Team-specific pipelines, boards and metrics could be maintained independently while shared resources (e.g. common libraries and templates) remains accessible at the group level. This structure proves effective in the coordination of front-end, back-end and QA teams working on the same sample project. GitLab supports epics, milestones and issue boards, all of which are important for coordinating work across multiple teams. User stories could be assigned to different teams via labels and assignees, but tracked collectively through a shared milestone and a parent epic. This approach is also endorsed in the book by Cowell et al.[44].

Although each team had its own issue board, the shared epic view allows project managers to see progress across all teams at a glance. This approach is similar to LeSS practices, in which a shared backlog is divided among multiple Scrum teams. However, LeSS also requires tight coordination and a minimal managerial hierarchy.

While GitLab is not explicitly designed for SAFe, as discussed in previous sections, its features can be adapted to SAFe's layered structure. For instance, GitLab's milestone system can be used to simulate Program Increments (PIs), with groups and projects representing different Agile Release Trains (ARTs). Nevertheless, GitLab does not natively provide SAFe-specific elements, such as PI planning boards or value stream views. Furthermore, Its flexible data structures (epics, labels, and milestones) allow SAFe concepts to be modeled relatively easily. Third-party plugins or customization is necessary.

Scalability also requires teams and stakeholders to maintain traceability of work and visibility of progress. GitLab provides group-level analytics, burn-down charts and epic roadmaps to support this. However, GitLab's enterprise-level reporting capabilities are more limited than those of dedicated portfolio management tools (e.g. Jira Align) according to Cowell et al.[44]. Plugins or external tools (such as GitLab's Premium or Ultimate tier analytics or custom Power BI dashboards via the GitLab API) may be necessary in order to gain more support for SAFe.

In summary, GitLab demonstrates support for scalability through group and subgroup structures enabling team and organizational partitioning. It also has epics, milestones and shared boards for multi-team coordination.

4.1.4 Regular project management tasks

This section uses the development of a To-Do List application to examine GitLab's standard project management features. The aim is to evaluate whether GitLab provides the necessary tools for communication, documentation, planning and non-technical collaboration — all of which are typical of many project management workflows. It also discusses issue tracking and ticketing functions, despite these being closely related to Agile. Nevertheless, they are also an essential component of traditional project management.

The main feature of GitLab's project management system is its issue tracker, which enables users to create, categorize, assign, and track tasks. During the development of the To-Do List application, tasks were organized using labels, milestones and boards. GitLab supports Kanban-style boards, which enables project managers and developers to visualize work in progress and plan upcoming iterations.

Tasks can be grouped under epics and milestones. These offer a hierarchical structure for larger planning needs. While GitLab's issue boards are less customisable than dedicated project management software, they are effective in tracking

project requirements, setting priorities and managing task statuses (e.g. 'To Do', 'In Progress' and 'Done').

Documentation is a central component of traditional project management, especially for onboarding new recruits, defining requirements, and project handovers. GitLab includes a built-in wiki for each project. It is used to document user stories, architectural decisions, feature definitions, and meeting notes. GitLab does not have built-in chat or video conferencing features. However, it does support comment threads on issues, merge requests and commits. These can be used to coordinate tasks, provide feedback and hold discussions in the projects.

GitLab also integrates with external communication platforms such as Slack and Microsoft Teams. This enables automatic notifications to be sent when code is pushed, issues are updated or deployments occur. These integrations enable a continuous feedback loop between developers and project stakeholders, supporting remote and hybrid collaboration. Nevertheless, teams often need to use third-party tools for real-time communication and centralized discussion spaces. GitLab is most effective when used within a broader communication ecosystem.

GitLab allows users to attach files to issues and wiki pages, which is useful for sharing diagrams and requirements documents. GitLab's repositories can host binary files and other non-code assets alongside the codebase. Project teams can use repositories (e.g. a /docs folder) for structured file storage and version control, or they can use external tools such as SharePoint or Google Drive. Links to these tools can be referenced in the GitLab Wiki or issues.

Project governance is supported through user roles, including Guest, Reporter, Developer, Maintainer and Owner. These roles define access levels for editing, commenting, and configuring project settings. This ensures control over project workflows.

GitLab was found to be useful for managing small- to mid-sized software projects,

eliminating the need for external tools. For teams in which software development tasks outweigh managerial coordination, GitLab provides sufficient ticketing functionality to support light task management alongside version control. For more complex or large-scale operations, GitLab can be integrated or used alongside with traditional project management tools to improve planning, reporting, and real-time communication. However, this sample project could not simulate a large-scale project.

4.2 Jira

Jira is a versatile project management and issue tracking tool. It was originally made as an issue tracking and project management tool by Atlassian[40]. Over the years it has become a useful tool for many different project types. Jira is made for agile and there are many things integrated into it that support agile and flexible software development. [45] This sample project was made with Jira premium free trial.

A sample project was created using Jira's automated "Scrum" -template. This template gives the basic structure for a Scrum team to operate on. Jira has also other templates for different projects and needs such as "kanban", "bug tracking", "budget creation", "project management and many other templates. These include many templates from software development projects, finance projects and marketing projects. Chhaya further explains these templates in a book "Ultimate Agile Administration with Jira"[40]. Some of these templates have similar base layouts and features. For this sample project, however, the 'Scrum' template was chosen. It has features relating to Scrum teams and agile software development, as well as other add-ons that support code collaboration. Users can create an empty template and add their own functions and features if they wish to do so.

Key features of this template include:

- Sprint Planning Tools: Pre-configured backlogs and sprint cycles facilitate iterative development.
- User Story Management: Native support for epics, user stories, and tasks ensures traceability from requirements to execution.
- Agile Reporting: Burn-down charts, velocity tracking, and cumulative flow diagrams are integrated for performance analysis.

To gain a better understanding of how Jira works as a tool, a set of tasks, user stories, timestamps, dependencies and sprints were created. These can be visualized using a variety of tools and functions. Below is examples of some of these visualization tools and functions.

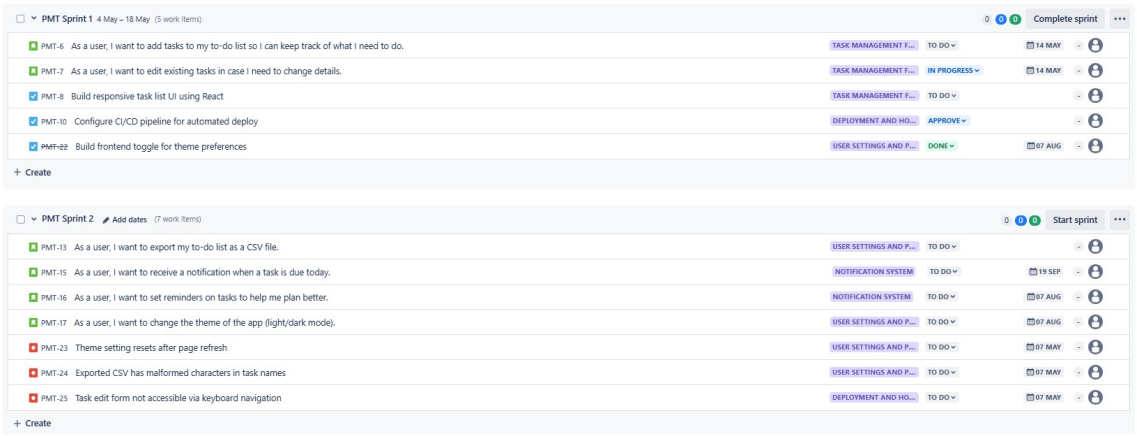


Figure 4.5: Jira backlog from a sample project

Figure 4.5 shows the backlog for prioritizing tasks for upcoming sprints. This backlog includes user stories, tasks, bugs and other project-specific items. It supports agile methodologies, enabling users to create and start sprints and customize them fully, whether in relation to time, people or the material in the backlog itself.

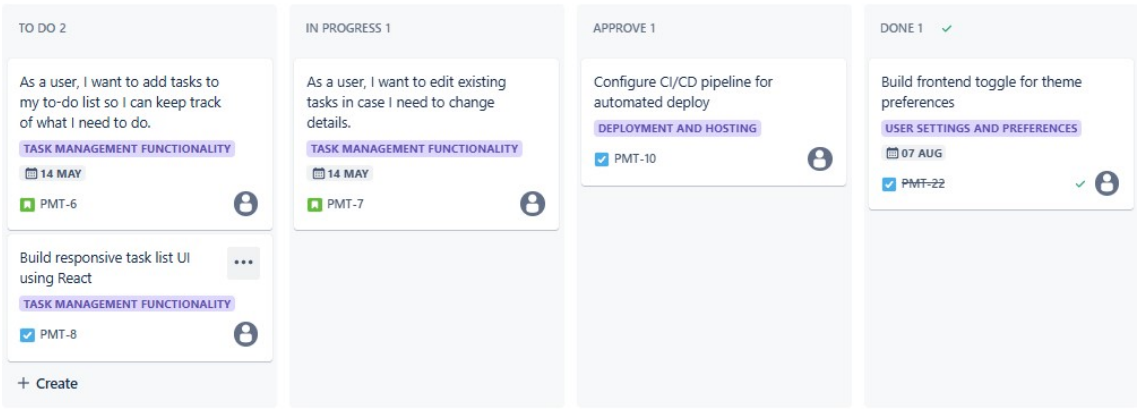


Figure 4.6: Jira sprint board

As shown in Figure 4.6, the sprint board is an important part of Jira's functionality. It provides a dynamic representation of task progression. Serving as both a monitoring tool and a coordination tool, it facilitates team alignment, task prioritization and iterative delivery. The board is organized into four distinct columns, each of which corresponds to a particular stage of the task's status within the sprint lifecycle such as:

1. **To Do**: Tasks awaiting initiation, such as *"Build responsive task list UI using React" (PMT-8).*
2. **In Progress**: Actively worked items, e.g., *"As a user, I want to edit existing tasks" (PMT-7).*
3. **Approve**: Tasks pending review, such as *"Configure CI/CD pipeline for automated deploy" (PMT-10).*
4. **Done**: Completed items, e.g., *"Build frontend toggle for theme preferences" (PMT-22).*

Each task card on the sprint board contains metadata that improves traceability and understanding of the context. This metadata usually comprises user stories

written as “As a user, I want to...”, categorization tags (e.g. “Task Management Functionality”), the name of the responsible user and timestamps. This supports transparency and accountability. It aligns with the Scrum and Kanban frameworks by facilitating continuous monitoring and the identification of workflow problems, while ensuring task ownership across development stages.

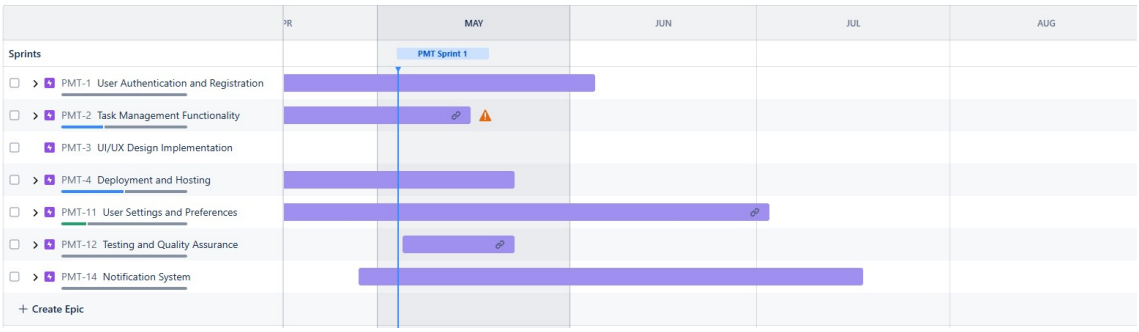


Figure 4.7: Jira timeline

The timeline view as shown in figure 4.7 provides a long perspective by mapping epics (e.g, *"User Authentication and Registration"*) across multiple sprints and calendar months. This view could be useful for release planning and milestone tracking, especially in environments practicing scaled Agile or in larger projects. It enables teams to manage dependencies, adjust delivery expectations and align feature implementation with strategic objectives over extended timeframes.

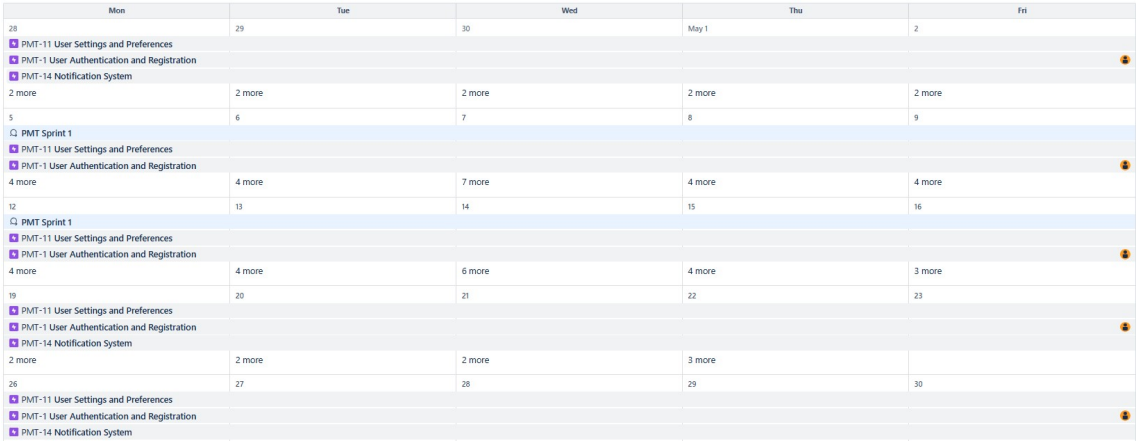


Figure 4.8: Jira calendar

As shown in Figure 4.8, the calendar view complements the timeline by offering a macro-level overview of sprint schedules, release windows and planned activities. It provides an overview of upcoming deliverables and deadlines in calendar format. This feature offers a different calendar-style view, which may be useful in some cases compared to the timeline view.

In summary, Jira offers a variety of visualization options to meet different needs and requirements. This is essential for scaled projects and environments with multiple teams.

4.2.1 Project management methodologies support

Jira is made for Agile project management as it was originally an issue tracking tool[40]. It offers specialized project templates for both Scrum and Kanban. It also offers comprehensive support for sprint-based workflows, backlog managing, user story tracking and reporting.

In the To-Do List application, a Scrum-based project was used for planning, executing, and monitoring iterations. The product backlog was created using Jira issues to categorize user stories, bugs and tasks with custom issue types. Sprints

were defined and managed via the Sprint Board. Jira also includes Agile reports, such as burndown charts, velocity charts, and cumulative flow diagrams. These reports provide insights into team performance and sprints. This strong support for Agile makes Jira especially well-suited to iterative and incremental development processes. Its flexibility enables it to adapt to specific team configurations.

Although Jira itself is not a DevOps tool, it supports DevOps workflows by integrating with CI/CD tools and source code repositories. In this sample project, Jira was integrated with GitLab. GitLab managed code related tasks and CI/CD pipelines. This integration enabled commits, branches and merge requests to be linked directly to Jira issues. This provided traceability from a user story to the exact code implementing it, which is a requirement in DevOps. Additionally, Jira's automation engine was configured to transition issues based on Git events. For example, an issue could be moved from 'In Progress' to 'In Review' when a merge request was opened. This automates some of the team's processes, making the workflow smoother and faster.

Although Jira lacks native DevOps pipeline functionality, it can be effectively integrated with tools such as BitBucket Pipelines, GitHub Actions and GitLab CI/CD. This enables Jira to be incorporated into DevOps toolchains. However, achieving a fully automated DevOps experience requires the use of third-party plugins and Atlassian ecosystem tools. This could be seen as a limitation in a real-world environment.

While Jira is not designed for the Waterfall model, it can be configured to support linear and traditional workflows. During the development of the To-Do List application, for example, the Waterfall model was simulated by creating a sequential issue hierarchy, with each development phase (e.g. Requirements, Development, Testing, and Deployment) represented as epics and tasks.

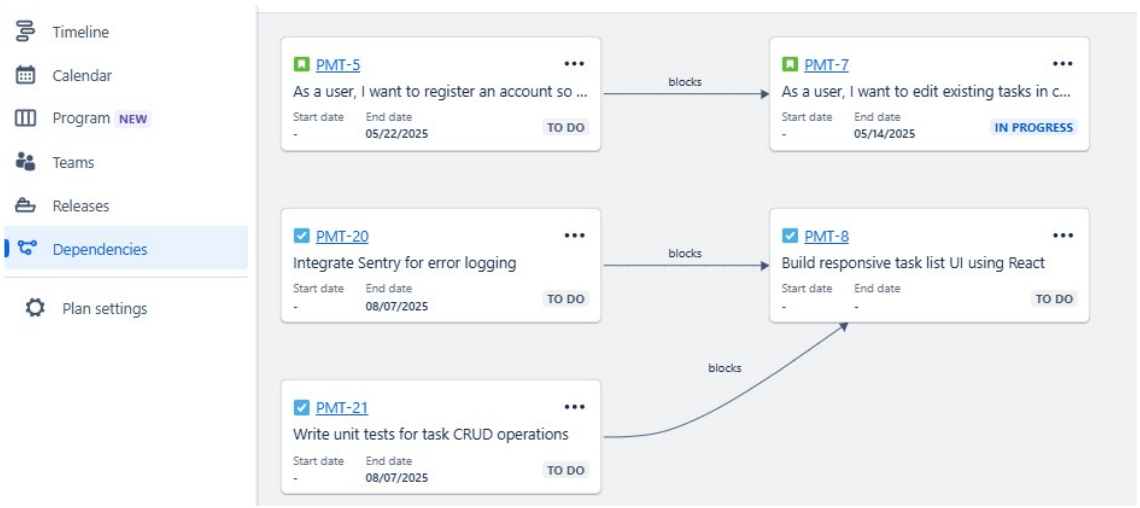


Figure 4.9: Jira dependencies

Advanced Roadmaps (available in Jira Premium) could be used to manage dependencies and timelines, and to visualize task sequences, target dates and critical paths (Figure 4.9). Gantt-style timelines can be created using plugins in Jira's premium products. However, Jira lacks some of the formal scheduling and budgeting tools available in Microsoft Project, which may limit its effectiveness in traditional project contexts. As discussed earlier, Jira has many templates. Due to time constraints, not every template could be tested. Therefore, there may be more suitable Waterfall-style templates available. According to the book by Chhaya [40], there are templates and plugins that support Waterfall when configured.

Jira supports SAFe through a combination of hierarchical issue structuring, portfolio management, and plugin-based enhancements. During testing, epics were used to group related user stories, and initiatives (available in Advanced Roadmaps) served as higher-level goals. Furthermore, Jira's multi-project boards, cross-team dashboards, and shared backlogs enable teams to collaborate on projects across multiple Agile teams. Jira integrates well with Confluence for documentation purposes and with Jira Align (a separate Atlassian product) for SAFe methods, including

PI planning, value streams, and Lean Portfolio Management according to Atlassian documentation[46]. In summary, Jira can support SAFe workflows, but often requires multiple Atlassian products or third-party apps.

Jira provides native support for Kanban, which was tested during the development phase of the To-Do List application. The Kanban board interface enables users to define custom columns representing workflow stages and providing a clear visualization of work in progress (WIP).

WIP limits can be configured on a per-column basis to enforce team policies and identify bottlenecks, which is important in Kanban. Jira's real-time updates and filtering options make it well-suited to teams that adhere to Agile and continuous delivery principles. Overall, Jira seems effective in supporting Kanban workflows.

In conclusion, Jira offers support for Agile, Kanban, and DevOps-related workflows, making it particularly well-suited to iterative, collaborative, and traceable software development environments. Waterfall and SAFe are also supported. However, more configuration and integration is required, especially for enterprise-scale implementations.

4.2.2 Code collaboration

Although Jira was originally developed as a project management tool, it has evolved into a central platform for code collaboration and a central hub of project related activities. It closely integrates with version control systems, continuous integration/continuous deployment (CI/CD) pipelines and code review tools. This section explores how Jira facilitates code collaboration and version control.

It is important to note that Jira does not support code collaboration natively. Its strength in this area stems from its integration with source code repositories such as GitHub, GitLab, and Bitbucket. These integrations allow developers to link commits, branches and pull requests directly to Jira issues. This creates a rather

seamless connection between development and project tracking. Consequently, developers, project managers, and other stakeholders can use the same software to track the project. In this section, GitLab was chosen as the tool to integrate with Jira. The reason for this is that GitLab is also the other tool that this thesis focuses on.

GitLab can be integrated with Jira. This is relatively easy to configure. There are two ways to connect: natively via the applications or by creating a Jira webhook. Once integration is complete, developers can add a GitLab project to a Jira project, thereby connecting Jira issues with GitLab issues and commits. For instance, when a developer creates a branch from a Jira issue key, the branch is automatically linked to the relevant issue. This enables project managers and other team members to monitor progress without having to switch tools. The issue key must be included in the commit comments for the two to be linked. Furthermore, Jira users can configure development panels to display the current status of related code, including open pull requests and deployment statuses. This level of visibility is important for iterative collaboration, particularly within Agile or DevOps teams, where feedback loops are short. These functions are designed for stakeholders, developers and managerial positions to oversee the project.

Jira supports pull requests and code reviews through its integrations. Users can carry out reviews, assign reviewers and track comments all within Jira. Therefore, it is not necessary to switch applications for these tasks. When a pull request is created, it can be linked to the relevant Jira issue and its status updated via operations performed within the code. For example, an issue might automatically change from 'in progress' to 'approve' when a pull request is opened and then to 'done' when it is merged and passes CI/CD. This automation ensures consistency in the development workflow and reduces the need for manual status updates.

From a collaborative standpoint, this single-program approach with the project

management and code collaboration integrations improves communication between developers, testers and managers. All stakeholders have immediate visibility into the working process. It therefore supports cross-functional collaboration, enabling testers to reference the exact code changes associated with a feature or bug fix.

Jira facilitates structured branching methodologies, including Gitflow and task-based workflow, through its code collaboration functionalities. Gitflow is a branching model used for code collaboration in Git. It enables teams to work on multiple branches simultaneously. This allows them to code separately and merge their code when they are ready. This topic was covered in a separate GitLab section. However, Jira appears to offer an integrated function for tracking issues and tasks.

```
stages:
  - deploy
  - notify

update-jira:
  stage: notify
  script:
    - curl -D- -u $JIRA_USER:$JIRA_API_TOKEN \
      -X POST \
      --data '{"body": "Deployed to staging."}' \
      -H "Content-Type: application/json" \
      $JIRA_API_URL/issue/$JIRA_ISSUE_KEY/comment
```

Figure 4.10: Jira and GitLab CI/CD integration

Jira can be integrated with continuous integration and delivery (CI/CD) tools such as GitLab CI/CD, Bitbucket Pipelines, and GitHub Actions. Jira should be able to automate workflows in response to code-related events. However, during testing, there was poor co-operation between Jira and Gitlab CI/CD. GitLab CI/CD and Jira do not natively support each other. Nevertheless, GitLab CI/CD can still be used with Jira (see Figure 4.10) via a webhook or third-party plugins. For example, a merged pull request could trigger an automated deployment to a staging environment and then update the relevant Jira issue to reflect the deployment status.

Jira also has an automation rules engine. This function can be coded to automate features based on Git events according to Atlassian website[46].

Jira CI/CD support	
Gitlab	Only through Webhook / API / Plugins
Github	Built-in support
Bitbucket	Built-in support

Table 4.1: Jira CI/CD integration

Jira integrates well with Bitbucket pipelines (Table 4.1). Github actions also integrated without the need to use Webhook or other plugins. This test was made only to figure out if they work or not. GitLab had some problems regarding integrations so a decision was made to briefly compare GitLab, Github and Bitbucket.

Maintaining traceability between code modifications and originating requirements is important in some projects. Jira has bidirectional linking of commits, pull requests and deployments to specific user stories or tasks[46]. This ensures end-to-end traceability throughout the project. Integration with code analysis tools such as SonarQube, Snyk and Checkmarx also enables teams to display code quality metrics, security vulnerabilities and technical debt directly within Jira. SonarQube was integrated with Jira via a third-party plugin in this setting. It seems that many code analysis tools require integration with third-party plugins.

Jira issues act as centralized collaboration hubs, allowing developers, testers, designers, product owners and other stakeholders to share feedback and files. These could be code snippets, messages, ratings or screenshots. The activity feed for each issue records all events, including status transitions, code commits, peer reviews, test outcomes and deployment milestones. This timeline provides stakeholders with a unified and comprehensive view of progress. For Agile teams, this transparency

helps create discussions in real time during Scrum ceremonies, such as sprint reviews and retrospectives.

However, Jira's effectiveness for code collaboration depends on proper configuration and keeping up to date with Jira and its integrations. Furthermore, some teams may find Jira's interface overly complex compared to more lightweight alternatives, such as GitHub Issues or GitLab Issues. Integrating these tools also introduces extra effort and complexity to the work process. Jira also lacks native capabilities for direct code editing or repository navigation. This may lead to the realization for smaller teams that configuring Jira may not always be worth it.

4.2.3 Scalability

Jira's project-based structure, combined with issue hierarchies, permission schemes and shared configurations, supports scalability. In the To-Do List application, multiple teams were simulated using separate Jira projects, each of which was configured with team-specific boards, workflows, and permissions. These projects were linked under a shared product roadmap using Advanced Roadmaps. This provides a view of goals and dependencies across teams.

This kind of architecture enables parallel development while maintaining organizational alignment. Permissions and roles could be defined per project. It enables secure team separation. This structure is well suited to both multi-team Scrum and program-level planning, which are core elements of LeSS and SAFe.

Although Jira does not offer clear support for LeSS, it can be configured to support its principles. In a small implementation, the LeSS approach was approximated by creating a shared backlog using a unified project, and then using labels, components, and teams to assign work to different Scrum teams. Additional tools, such as Advanced Roadmaps, may be necessary for larger-scale LeSS implementations.

Jira offers tools to support SAFe adoption, particularly when used alongside

Jira Align, Advanced Roadmaps, or third-party plugins such as BigPicture. During testing, SAFe's layers (team, program, solution, portfolio) were simulated using a combination of issue hierarchy levels:

- Initiatives and epics at the program and portfolio levels.
- Stories and tasks at the team execution level.
- Milestones and versions used to simulate Program Increments (PIs) and Releases.

Jira's Advanced Roadmaps gives visibility across multiple teams and allows linking of lower-level issues (e.g., tasks) to higher-level goals (e.g., epics). Dependencies could be visualized between issues, helping identify cross-team blockers. Simulated teams were assigned to separate boards and could operate independently while maintaining alignment via shared epics and milestones. Jira Align provides native support for SAFe concepts like value streams, ARTs (Agile Release Trains) and PI planning, but requires additional licensing and setup under the Atlassian ecosystem[46].

Jira's scalability is further enhanced by its flexible board configuration and JQL (Jira Query Language). During testing, cross-team dashboards were created using filters that represented issues from multiple projects. This provided real-time visibility of work progress, team velocity and sprint burndown across all the development teams working on the To-Do List application. Stakeholders could monitor metrics such as sprint velocity and issue throughput without having to navigate to each project individually[46].

Although Jira appears to be highly scalable, its efficiency depends on proper configuration. As the number of teams, boards and issues increases, system performance and usability may be impacted. Careful management of custom fields, workflows and notifications is essential to prevent administrative oversight. This

makes maintaining the project management software somewhat challenging. This is especially difficult if the team is small and there is no one to specifically operate and customize Jira.

Additionally, the Standard version of Jira has limited support for cross-team visibility and scaled planning. Features such as Advanced Roadmaps, issue hierarchy beyond epics and multi-project planning tools are restricted to the Premium and Enterprise tiers. However, non-commercial users can access free trials.

In summary, Jira offers scalability features, making it suitable for scaled teams and organizations using: LeSS-style shared backlogs and coordinated Scrum teams; SAFe-style portfolio and program-level planning via epics and initiatives; Cross-team visibility through shared dashboards and roadmaps; Custom workflows and granular permissions for team separation.

In conclusion, Jira scales from a single Scrum team to a multi-team program structure relatively easy. While Jira supports both LeSS and SAFe in principle, full implementation requires Premium-level features or integration with Jira Align.

4.2.4 Regular project management tasks

This section evaluates Jira's capacity to support general project management functions such as communication, planning, documentation and collaboration through the practical development and coordination of a To-Do List application. It also takes a look at the issue tracking system that Jira was originally made for.[47]

Jira's core functionality lies in its adaptable issue tracking system, which can be used for a wide variety of project types and methodologies. Jira issues are highly customizable. These can be tasks, stories, bugs, epics or sub-tasks, for example. In the To-Do List project, tasks were broken down into smaller deliverables and tracked using custom workflows, priorities and assignments.

Jira facilitates asynchronous communication through issue comment threads,

mentions and activity logs. This functionality enables team members to coordinate work, ask questions and provide feedback directly on tasks.

Jira natively integrates with communication platforms such as Slack, Microsoft Teams and email, enabling automatic notifications about updates to issues, comments or workflow changes. This integration supports remote and hybrid collaboration models, both of which were simulated in the application project. Jira also integrates with ERP systems like Microsoft Dynamics and SAP S/4HANA. These integrations widen the use cases of Jira. Next, this thesis will take a closer look at how Jira integrates with Slack.

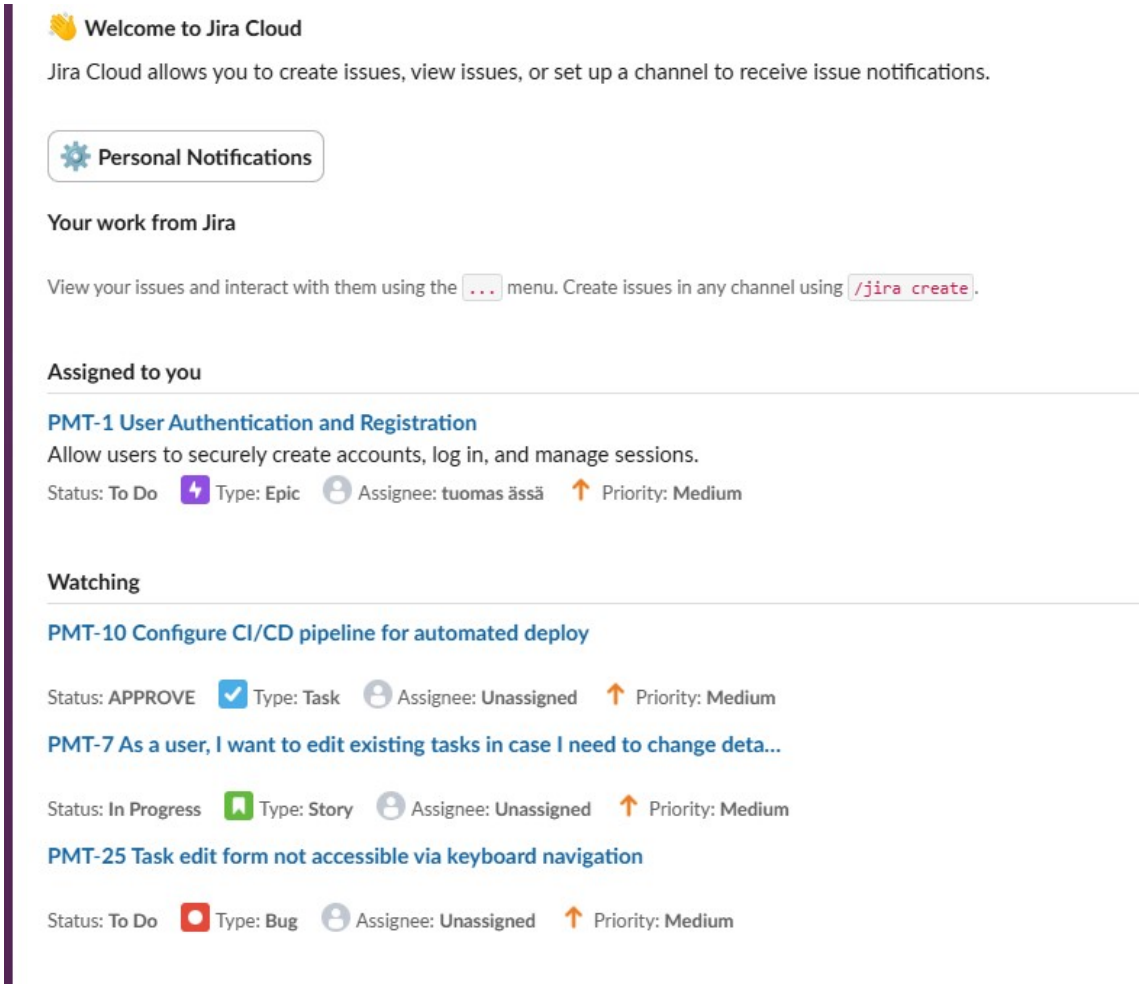


Figure 4.11: Jira integrated with Slack. The view from Slack app.

Slack is a messaging tool for software development similar to Teams or Discord. Its main purpose is communication and collaboration. It can also be integrated into Jira. Key Features of Slack-Jira Integration are real time notifications, issue creation and management, access to work items and collaboration.

The integration allows teams to receive instant notifications in Slack about Jira issues, such as status updates, assignments or comments. This ensures that team members stay informed without needing to switch contexts between platforms. For example, users can view issues assigned to them or those they are watching directly within Slack (Figure 4.11). These include details like status, type and priority. Slack users can create Jira issues directly from any channel using slash commands (e.g., /jira create). The user can make simple tasks without opening Jira speeding up the workflow.

The integration provides a consolidated view of Jira issues within Slack, including epics, stories, tasks and bugs. For instance, figure 4.11 shows an epic ("User Authentication and Registration") and a bug ("Task edit form not accessible via keyboard navigation"), each with metadata like assignee, priority and status. This updates in real time whether changes are made anywhere else. Teams can discuss and resolve issues in the same environment where communication occurs.

While Jira does not offer real-time messaging or video conferencing features internally, it functions effectively as a central discussion hub around tasks and projects. Each issue acts as a self-contained record of decisions, comments and attachments. Though Jira does not include more advanced documentation features natively, it integrates with Confluence which is Atlassian's dedicated documentation tool[46]. This tool was not tested in this thesis.

Jira provides multiple options for project planning and scheduling, particularly in other Atlassian tools and Jira Premium tiers. Milestones can be represented through versions or fix versions, while timelines are visualized using Advanced Roadmaps.

In the To-Do List application, sprint planning and milestone tracking were executed via the backlog and sprint board. Workload balancing and velocity tracking were visualized using burndown charts, cumulative flow diagrams and team reports. Jira also includes goal-setting features, such as Objectives and Key Results (OKRs) through third-party apps which align with broader organizational planning frameworks. OKR-board and Workboard are these kind of plugins. OKR-board plugin was tested in this sample project briefly. It seemed to work, however further usability could not be tested.

Jira supports file attachments in all issues, making it easy to share screenshots, documents and data files. In the context of the To-Do List application, assets like UI prototypes and test case spreadsheets were attached directly to relevant issues. Jira integrates with cloud storage platforms such as Google Drive, OneDrive and Dropbox allowing documents to be shared as links or embedded previews. Google Drive was linked to the sample project. It seemed to work well linking contents from Google Drive to Jira Issues. This could be useful if larger sized files are needed in the project.

In summary, Jira is a versatile platform that extends well beyond code tracking to support regular project management activities, including:

- Issue tracking with custom workflows and visual boards
- Asynchronous collaboration and communication via comments and notifications
- Integration with Confluence for documentation and knowledge sharing
- Planning and forecasting tools such as burndown charts and advanced roadmaps
- File management and third-party app support
- Granular permissions and workflow configurations for structured collaboration

In the To-Do List application, Jira proved effective in coordinating tasks, managing timelines and documenting progress in a centralized app. While its communication and document editing capabilities rely on integrations, its modular architecture and large ecosystem offer flexibility for diverse project environments.

Jira has thousands of tools and plugins that it can integrate to [47]. Users can even create their own plugins if needed. In this thesis only few of these plugins were tested. These integrated tools seem to be extremely effective and the potential for automation is significant. However, this comes with a cost. These are quite hard to implement and automate. Finding right and suitable solutions and workflows seemed also hard. There are so many options and hybrid solutions which makes it hard to find the right, useful and seamless combinations.

4.3 Microsoft project

In order to evaluate the capabilities of Microsoft-based ecosystems, this thesis involved using Microsoft Project to manage the implementation of a To-Do List application as a sample project. Microsoft Project is primarily a tool for project planning, resource scheduling, and timeline visualization. In contrast, GitLab and Jira integrate directly into the software development toolchain. Microsoft Project, on the other hand, operates at a more abstract level. It offers high-level planning functionalities suited to traditional project management methodologies, such as the Waterfall model, or hybrid models. In this case, GitLab was used as a version control tool, whereas Microsoft Project was the separate project management tool slightly similar to Jira. The Microsoft ecosystem offers a wide range of tools that can assist with project management. Microsoft Project can be used alongside Project for the Web, Project Online, Azure DevOps, Planner, SharePoint, Power BI, and Project Roadmap. Each of these platforms supports specific aspects of project, program or portfolio management. GitHub is also owned by Microsoft and it can be integrated

into Azure DevOps. Functionally, Azure DevOps may be regarded as a slightly different and light-weight alternative to Jira, whereas GitHub is often positioned in close comparison to GitLab.

The sample project in Microsoft Project is created differently since it has significant differences compared to the other tools. The project was initiated by defining the overall scope of the To-Do List application. The functional requirements were gathered and translated into a list of epics, such as: 'As a user, I want to add tasks to my to-do list so I can keep track of what I need to do' and 'As a user, I want to be able to edit existing tasks in case I need to change details'. This strategy was chosen because it is similar to the implementations made in Jira and GitLab.

These deliverables were broken down into work breakdown structures (WBS) using Microsoft Project. This provided a clear, hierarchical overview of the tasks required to complete the application. Each WBS item was then broken down further into smaller, more manageable packages representing tasks in Jira and GitLab (e.g. 'Build responsive task list UI using React', 'Configure CI/CD pipeline for automated deployment', 'Create release and deployment plans').

Once the task list had been finalized, an estimated duration was assigned to each activity. Tasks were sequenced using logical dependencies (finish-to-start and start-to-start). For example, 'Configure CI/CD pipeline for automated deployment' depended on 'Make release and deployment management plan'. These dependencies were visualized in a Gantt chart, which allowed the project schedule to be updated in real time as tasks progressed or potential delays occurred.

The hypothetical project team was defined in the Resource Sheet in Microsoft Project, with roles including front-end developer, back-end developer, tester, and product manager. Each task was assigned one or more resources, and their availability was specified in terms of working hours per day and calendar exceptions (e.g. weekends). Microsoft Project automatically adjusts task start dates based on

resource constraints and dependencies. This helps to maintain a balanced workload.

Actual work progress was tracked using manually entered completion updates. As tasks were marked as complete, Microsoft Project updated the timelines and recalculated the project finish date based on the actual data.

Microsoft Project has a milestone feature which could be similar to sprints in agile setting. This thesis focuses on agile and other software development methodologies so the plan was to force agile into the applications if applicable. However, this feature is probably not well suited for this kind of use. Few milestones were created as a test such as "sprint 1", "project completed" and "release and deployment management is ready". However, this did not work out as planned. These milestones are used to monitor progress and assess whether the project was on track similar to sprints. At least creating a similar project setting related to Jira or Gitlab was hard.

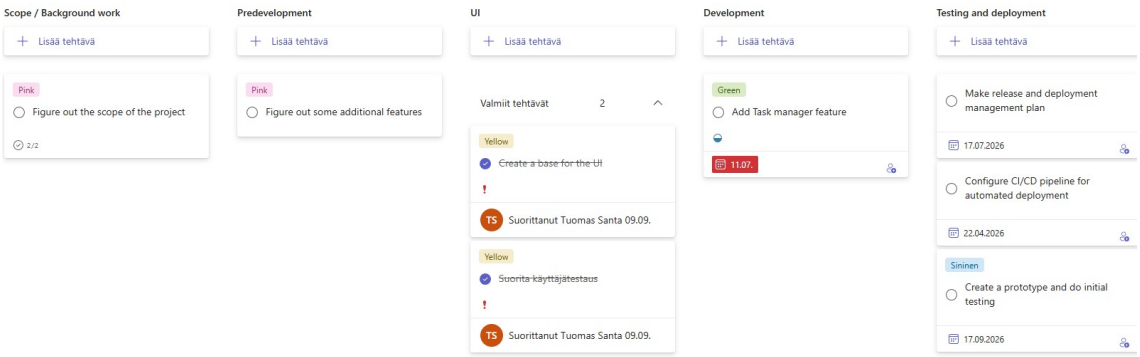


Figure 4.12: Microsoft Planner

Figure 4.12 shows how Planner allocates work and visualizes it. Planner is another Microsoft product that offers similar solutions to Project for the web which is being replaced by Planner. Planner can integrate into the Microsoft toolchain. This integrates Agile specific managerial tasks and a communication platform to one tool. It has software project management templates. It can track tasks and show some visualizations similar to Jira. However, Planners capabilities are significantly

more limited compared to Jira. It can be integrated into Teams and some other Microsoft products offering light project management and project analysis tools. It offers some kind of Kanban boards, however the functionalities are somewhat limited at least in the student version. Planner can further complement Microsoft Projects functionalities in software development settings.

4.3.1 Project management methodologies support

Microsoft Project has traditionally been designed around the Waterfall model. For this test, however, the project was initially structured using Agile. This may not be the most suitable approach for Microsoft Project. However, in this section, we take a closer look at how it works with other methodologies.

Waterfall principles could be used in Microsoft Project by breaking the process down into sequential phases: Requirements, Design, Implementation, Testing and Deployment. Each phase could be planned using tasks and subtasks and organized in a hierarchical work breakdown structure. Dependencies are defined through predecessors and successors, enabling the use of Gantt charts to visualize timelines. Resources, such as development hours, are allocated to each task, and the schedule is adjusted using baseline and actual comparisons. This allows for variance tracking and earned value analysis. This is understandable, given that Microsoft Project was designed with the waterfall model in mind.

Although it was not originally designed for Agile, Microsoft Project has introduced features that enable some Agile workflows. Agile principles were simulated by creating sprints with individual development tasks nested beneath them. Task boards allowed columns such as “To Do”, “In Progress” and “Done” to be used. However, Microsoft Project’s Agile functionality is more limited and less intuitive than tools that are specifically designed for Agile workflows (e.g. Jira or GitLab). However, Microsoft Planner and Azure DevOps supports Agile workflows better.

Microsoft Project also has some templates for Agile and Scrum teams. These templates can handle the managerial side of Agile development. However, those lack tools and functions that the developers need.

A notable limitation is the lack of support for real-time team collaboration and continuous updates. Although burndown charts and task boards are available, they require configuration and are not closely integrated with code or development tools. This makes Agile in Microsoft Project more administrative than operational. However this is probably intentional since Microsoft owns GitHub and Azure DevOps which can handle the operational and developer centric tasks.

Microsoft Project is not built to support DevOps workflows. Unlike GitLab, it lacks native tools for version control, CI/CD, and environment automation. Consequently, DevOps in Microsoft Project must be managed indirectly, often relying on manual data entry or integrations with Azure DevOps or other platforms.

In the sample project, GitLab was used to manage CI/CD and code collaboration, while Microsoft Project only served as a high-level planning tool. In order to link development status to project progress, status updates had to be manually extracted from GitLab and entered into Microsoft Project. This created friction and reduced real-time visibility. Since Azure DevOps and GitHub are Microsoft products this would have been better with them integrated into the workflow.

Microsoft Project can represent DevOps-related tasks and events (e.g. testing and deployment) as part of the project plan. However, it does not support automation or direct integration with deployment workflows. Consequently, it is not well-suited to teams practicing DevOps, unless used alongside other tools since DevOps is almost entirely code related activities.

SAFe requires capabilities for portfolio-level planning, team-level execution, and coordination between Agile Release Trains (ARTs). Microsoft Project has some potential to support SAFe due to its resource planning, Gantt scheduling, and hier-

archical task structures. In this test, SAFe was represented using summary tasks as program increments, with features and stories listed as subtasks. Milestones were used to mark sprints. Microsoft Project also supports multi-project views, which can simulate multiple Agile teams working under a common portfolio. However, this experiment didn't seem to work based on brief testing.

SAFe adoption in Microsoft Project remains largely theoretical and manual or limited. The platform lacks specialized tools for PI planning, value stream mapping and Agile team coordination. While SAFe concepts can be modeled, they are not inherently supported by the tool and require high configuration effort. Microsoft Project has Agile templates which can be used in agile setting. However these still seem to lack some features needed for such use.

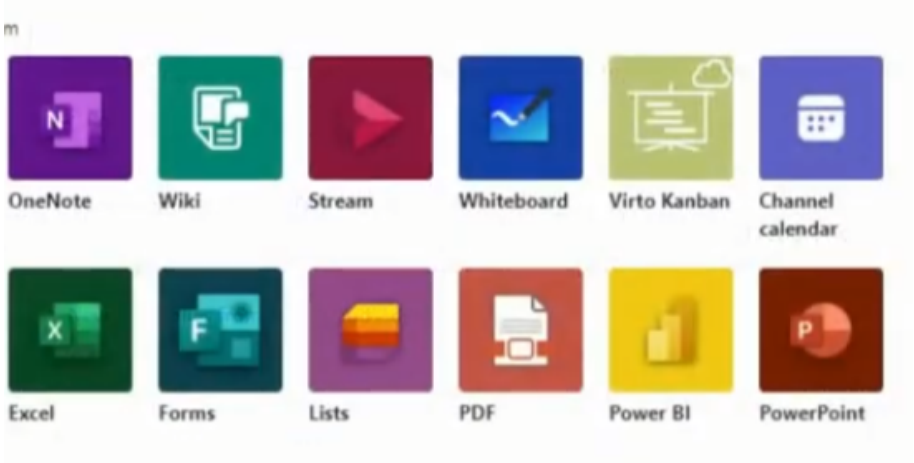


Figure 4.13: Microsoft Teams Kanban-style board

Microsoft Teams provides support for Kanban (figure 4.13). Users can create task boards that visually resemble Kanban boards via Virto Kanban or Planner for example. Azure DevOps has also its own Kanban boards. However, Microsoft Teams Kanban boards does not have similar functions compared to Jira or GitLab due to the coding aspect of those tools. Moreover, Microsoft Project's traditional scheduling logic (e.g., fixed dates and task durations) clashes with the fluid nature of Kanban. While a visual board may exists, its functionality is primarily cosmetic

unless linked with more agile-friendly tools like Azure DevOps.

In summary, Microsoft Project is best suited for Waterfall-based planning, offering support for linear task sequencing, scheduling and budget management. It has some tools to accommodate Agile and Kanban methodologies but lacks the full operational depth of specialized tools. Its ability to support SAFe is limited to structural modeling and visualization. DevOps integration is minimal, requiring external platforms like Azure DevOps or GitHub to manage CI/CD pipelines. Microsoft Project may function best as a strategic planning overlay, complemented by more flexible execution platforms.

However, Microsoft Project can still integrate to other tools. Microsoft's Project for the Web and nowadays Planner with Project Accelerator extends Microsoft Project's capabilities to better support Agile methodologies. These tools enable the use of Kanban boards, sprint planning, backlog management and epic tracking. They are built on the Microsoft Platform, which allows some integrations across the tools. They allow for a more flexible and modular approach to project management that aligns with Agile principles according to Microsoft Project documentation [48].

A key to enabling DevOps practices within the Microsoft Project environment is through integration with Azure DevOps. Microsoft provides official integration options that allow for the synchronization of tasks, sprint backlogs, user stories and work items between Microsoft Project and Azure DevOps. This enables DevOps style development. However, this could not be tested during this thesis since these tools have no free trials. This short analysis was based on these books by Soni [49] and Rossberg[50]. Since Microsoft owns these other products, they seem to integrate well according to Soni[49] and based on brief experiments.

Power BI is also a Microsoft product that can be utilized to visualize and analyze data from both Microsoft Project and Azure DevOps. This integration allows organizations to generate customized dashboards and reports, such as sprint progress,

burndown charts, and milestone tracking. These provide deeper insight into both Agile and traditional project metrics. According to Ciure's thesis from 2024 [51], Power BI is a great tool compared to Powerpoint when creating visuals for projects. Based on Ciure's [51] findings it seems to integrate well with Azure DevOps. Microsoft Project can also be configured to work with third party plugins. However, this requires more configuration effort and a question arises if larger corporations would prefer to add 3rd party plugins into Microsoft software.

4.3.2 Code collaboration

In this section this thesis takes a deeper dive on how - or whether - Microsoft Project facilitates code collaboration among team members. Unlike GitLab or Jira, Microsoft Project is primarily designed as a project planning and scheduling tool. It lacks native capabilities for version control, source code management or integrated CI/CD pipelines.

Microsoft Project did not support any form of direct code collaboration. As it lacked integration with Git repositories or development tools, all collaborative coding activities had to take place outside of the platform. In this sample project, GitLab was used for code reviews, branching and testing, while Microsoft Project was used only for high-level project planning. It was not possible to automatically link commits, branches or merge requests to tasks in Microsoft Project, which significantly limited its usefulness for tracking the progress of code-related work in real time.

Manual updates were therefore required. For example, once a feature was completed and merged via GitLab, the corresponding task in Microsoft Project had to be marked as complete manually. This workflow could cause delays and inconsistencies which opposes the Agile ideology.

Unlike Jira or GitLab, Microsoft Project did not offer issue tracking or ticketing

functionality that is linked to Git-events. This meant that any bugs discovered during development had to be tracked in a separate system, such as GitLab Issues or external spreadsheets, and then manually reflected in Microsoft Project as new tasks or delays. In the context of Agile or iterative development, this kind of workflow was slow and did not support the fast feedback loops needed for testing.

Furthermore, Microsoft Project lacks collaborative features such as inline commenting, code diffing and code review. Those working with the code could not exchange feedback on specific changes within Microsoft Project, and stakeholders could not visualize which code commits were associated with specific tasks or milestones. Despite these limitations, Microsoft Project could provide a valuable macro-level view of the project schedule. Gantt charts help to visualize dependencies and forecast delays when development bottlenecks occur.

In conclusion, Microsoft Project offered no native support for code collaboration. All activities related to source code management, review and CI/CD were conducted externally. While it is a powerful scheduling tool in traditional project management contexts, its application in collaborative software development is limited unless used with integrated development platforms like GitLab or GitHub. Azure DevOps can also be used for code related tasks. According to Soni[49], Azure DevOps integrates well with GitHub's code repository's.

4.3.3 Scalability

This section evaluates the scalability of Microsoft Project based on the development and coordination of a To-Do List application in various scenarios involving multiple teams and frameworks, such as LeSS and SAFe.

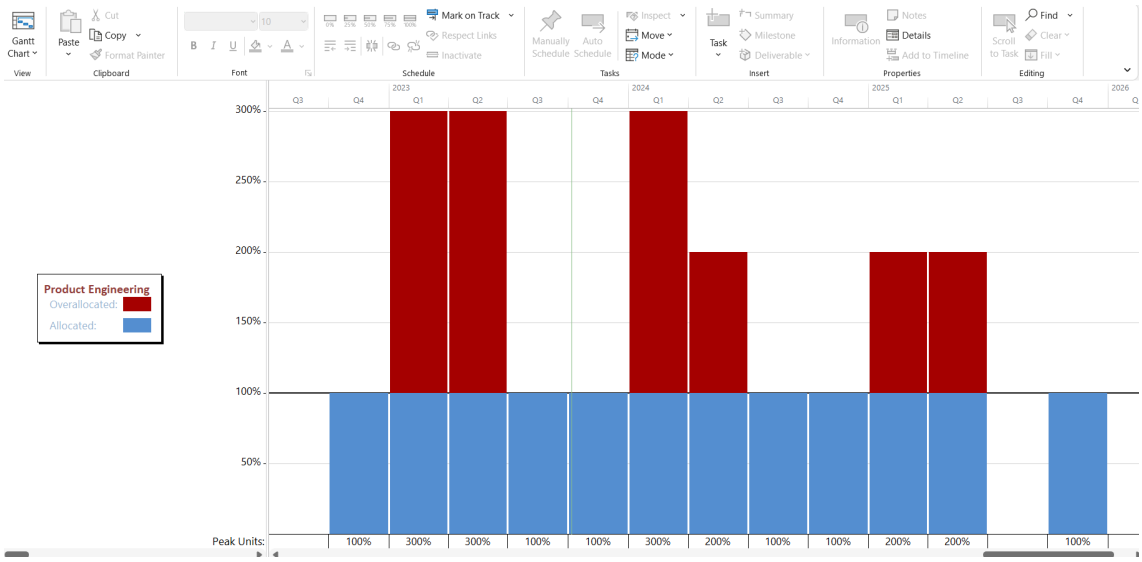


Figure 4.14: Microsoft Project resource allocation

One of the core advantages of Microsoft Project is its resource-centric model. This structure supports the allocation and tracking of resources across multiple tasks, teams, and timelines. In the To-Do List application, different hypothetical development teams (e.g. front-end, back-end and QA) were modeled as separate resource groups. Each team was assigned individual workloads and assignments. While these teams can operate under separate schedules, they can still align to overarching delivery dates and milestones through larger project plans. This model enables managers to maintain high-level visibility of dependencies and resource usage while allowing teams to manage their specific tasks using their own timelines. Resource allocation can be tracked quite well inside the Microsoft Project. Figure 4.14 shows some over allocated work in the project. This feature is useful for managerial positions.

Microsoft Project is not natively designed for scaled Agile frameworks such as LeSS. In this thesis a simulation of this approach was used to coordinate multiple Scrum teams working from a shared backlog. However, Microsoft Project's tradi-

tional waterfall-style planning logic (e.g., Gantt charts, fixed start/end dates) imposes a level of rigidity that contrasts with the iterative nature of LeSS. Therefore, Microsoft Project may be more suitable as a coordination tool for leadership, rather than a daily planning tool for agile teams. Azure DevOps can be implemented with SAFe according to Riti et al.[52].

The native toolset in Microsoft Project aligns more naturally with the hierarchical planning model used in SAFe. It could support: (1) strategic themes and portfolio epics at the enterprise level; (2) program increments and release trains modeled as nested summary tasks or sub-projects. (3) team-level sprints, represented as task groups or milestones.

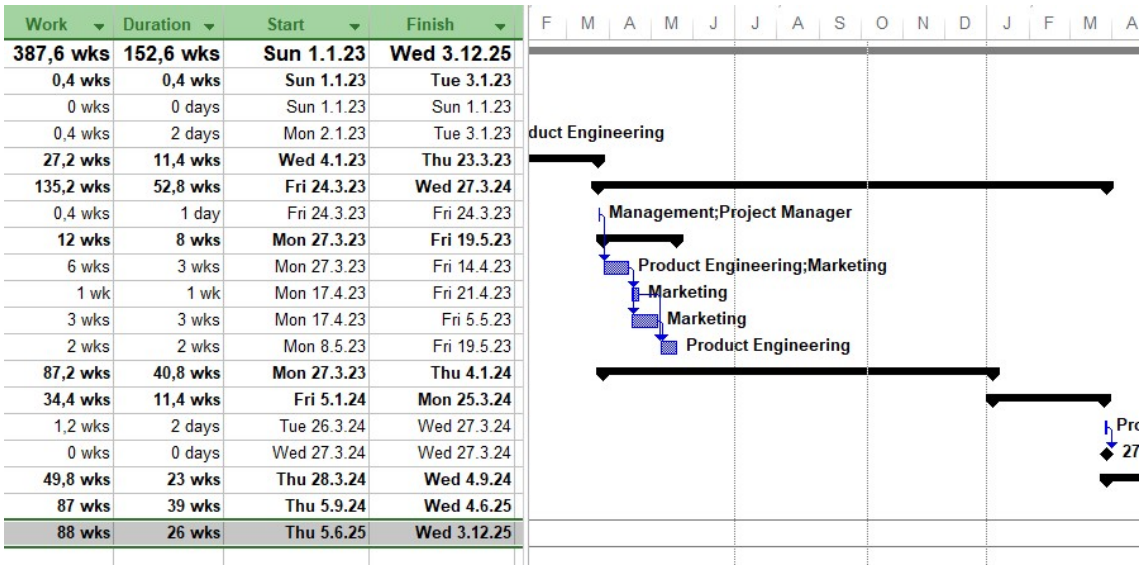
Despite its planning features, Microsoft Project has limitations with regard to developer centric agile collaboration and team-level execution. The interface is optimized for project managers rather than developers and lacks features such as sophisticated issue tracking, pull request integration and built-in CI/CD pipelines. However with tools such as Azure DevOps it can be more friendly towards developers.

In conclusion, Microsoft Project is most effective when used alongside tools that support team agility and code collaboration, such as Azure DevOps and GitHub. Changes from agile tools must be manually configured or integrated to be reflected in real time. Furthermore, the licensing requirements for full access to Project Online or the Power Platform may present challenges for smaller organizations. Microsoft Project could be used at a higher level, whereas Azure DevOps or Jira and GitLab or GitHub could be used for development purposes. However, integrating and automating these tools could be challenging or at least time consuming.

4.3.4 Regular project management tasks

This section evaluates how Microsoft Project supports standard project management practices when implementing a To-Do List application, with a focus on communication, coordination, documentation and planning. This is also done with Agile and other software development methodologies in mind.

Microsoft Project is centered around the Work Breakdown Structure (WBS) paradigm. During development of the To-Do List application, tasks were broken down into smaller work items, grouped under deliverables, and arranged hierarchically using phases and sub-tasks. Each task was assigned start and end dates, a duration, dependencies, and predecessor relationships. These details were used to create a Gantt chart that visualized the project timeline.



Work	Duration	Start	Finish
387,6 wks	**152,6 wks**	**Sun 1.1.23**	**Wed 3.12.25**
0,4 wks	**0,4 wks**	**Sun 1.1.23**	**Tue 3.1.23**
0 wks	0 days	Sun 1.1.23	Sun 1.1.23
0,4 wks	2 days	Mon 2.1.23	Tue 3.1.23
27,2 wks	**11,4 wks**	**Wed 4.1.23**	**Thu 23.3.23**
135,2 wks	**52,8 wks**	**Fri 24.3.23**	**Wed 27.3.24**
0,4 wks	1 day	Fri 24.3.23	Fri 24.3.23
12 wks	**8 wks**	**Mon 27.3.23**	**Fri 19.5.23**
6 wks	3 wks	Mon 27.3.23	Fri 14.4.23
1 wk	1 wk	Mon 17.4.23	Fri 21.4.23
3 wks	3 wks	Mon 17.4.23	Fri 5.5.23
2 wks	2 wks	Mon 8.5.23	Fri 19.5.23
87,2 wks	**40,8 wks**	**Mon 27.3.23**	**Thu 4.1.24**
34,4 wks	**11,4 wks**	**Fri 5.1.24**	**Mon 25.3.24**
1,2 wks	2 days	Tue 26.3.24	Wed 27.3.24
0 wks	0 days	Wed 27.3.24	Wed 27.3.24
49,8 wks	**23 wks**	**Thu 28.3.24**	**Wed 4.9.24**
87 wks	**39 wks**	**Thu 5.9.24**	**Wed 4.6.25**
88 wks	**26 wks**	**Thu 5.6.25**	**Wed 3.12.25**

Figure 4.15: Microsoft Project Gantt-chart

One of Microsoft Project's features is its scheduling engine, which automatically recalculates timelines when task dependencies or durations change. In the To-Do List project, major features and larger parts of the app were defined as milestones and linked to lower-level tasks in order to monitor delivery status. The timeline,

calendar and task usage views enabled tracking of both micro- and macro-level progress. Figure 4.15 shows the Gantt-chart in Microsoft Project.

Microsoft Project does not natively include modern chat or commenting capabilities on tasks. However, when used through Project for the Web or Project Online within the Microsoft ecosystem, it can integrate with Teams, Outlook and OneDrive to enable communication workflows according to Soni[49]. Microsoft Project does not include a built-in wiki or documentation function. However, this could be fixed using it alongside Microsoft OneNote and SharePoint. Microsoft also has CRP and ERP systems known as Microsoft Dynamics. These can also be useful tools for larger corporations.

In summary, Microsoft Project is a platform designed for traditional project management workflows. However, it lacks communication and documentation tools. Although it also lacks native tools for code management or Agile workflows, Microsoft Project is excellent at managing structured and deadline-driven projects. Its effectiveness increases significantly when used alongside Microsoft 365 products and Power platform tools. This enables a more integrated and collaborative project environment. As these ecosystems grow, there may be opportunities for further integration. Could Microsoft Project be seamlessly integrated with GitLab, GitHub, BitBucket and Azure DevOps? Microsoft Project has one huge advantage — it's part of the Microsoft ecosystem. This could enable Microsoft to create a larger ecosystem with features and plugins similar to those in Jira. Microsoft already has other products, such as Project for the web, Planner, Power BI, Excel, GitHub, Azure DevOps, Teams and Outlook. There could be a lot of room for automating processes.

4.4 Comparative summary of the tools

This previous testing evaluated three widely used project management tools: GitLab, Jira, and Microsoft Project. Each tool was assessed against four key dimensions: support for project management methodologies, scalability, code collaboration, and regular project management functionalities. The comparative analysis highlights the strengths and limitations of each platform based on the sample project and feature evaluation.

Each platform exhibits varying degrees of alignment with modern and traditional project management methodologies. GitLab provides extensive support for Agile, DevOps and Kanban practices. It can be scaled more effectively when integrated with Jira. Jira also supports Agile, DevOps and Kanban. Microsoft Project, on the other hand, focuses on Waterfall and structured planning methodologies. GitLab seems to have the basic toolset for software projects, but could benefit from external software. Similarly, Jira benefits from external software implementations, which can be integrated to help automate the process. Microsoft Project also has an excellent toolset within the Microsoft ecosystem. GitLab can integrate some tools, but its capabilities are significantly more limited than those of Microsoft ecosystem or Atlassian with Jira. Figure 4.16 combines all the findings from this empirical testing into one table.

	Atlassian Jira	Gitlab	Microsoft Project
License /Price	Free community licenses for open source and academic projects ; Multiple price points and free trials	Free version and multiple price points for commercial use	Multiple price points for commercial use and free trial
Platform	Web-based / installed	Web-based / Git	Web-based / installed
Agile, DevOps support	Support for agile; DevOps with integrations for CI/CD etc	Support for Agile and DevOps	Support with configuration / other Microsoft tools / Agile templates
Backlog management	Support	Support	Support
Epics and Issues	Support	Support with premium product	No direct support; Support with other Microsoft tools
Portfolio planning	Support with integrations / other Atlassian products	Some support with integrations / configuration	Support especially with other Microsoft tools
Task board visualizations	Support with many different visualizations	Support	Some support: Agile specific may be limited
Scalability / support for SAFe, LeSS	Support especially with integrations / other Atlassian products	Some support	Good for scaling traditional projects. Other Microsoft tools expand the possibilities
Code Collaboration, Version control	Support with integrations (e.g. GitLab)	Support	Support with other Microsoft tools
Code analysis	Support through third party plugins and integrations	Support especially with third party integrations	Support with other Microsoft tools and integrations
CI/CD	Support with integrations (e.g. GitLab)	Support	Support with other Microsoft tools (Azure DevOps)
Communication and documentation	Support with integrations/ other Atlassian products	Support with integrations (e.g. Slack)	Support with other Microsoft tools
Tracking resources, time, tasks	Support. More features with integrations / other Atlassian products	Support	Support, However Agile style may be limited

	Atlassian / Jira	GitLab	Microsoft Project
Multiple projects products, team	Support. More features with integrations / other Atlassian products	Support	Support
Support for waterfall (e.g. Gantt-charts)	Support when configured	Minimal Support	Support
Integrations with ERP systems	Integrates Microsoft Dynamics and SAP	No direct support	Microsoft Dynamics
Target users	Hybrid space for many professionals	Developer-centric	Project-manager centric. Microsoft has other hybrid and Developer centric tools
Kanban boards	Support	Support	Support with many different tools under the Microsoft ecosystem

Figure 4.16: Comparison of the tools based on previous findings

Scalability was evaluated in terms of tool performance and process coordination across multiple teams, departments, and enterprise sizes. The evaluation of scalability is hypothetical since simulation could not simulate all the nuances of software development. The findings were contrasted or supported with other material. GitLab scales well within a DevOps-driven environment. It supports multi-repository management, group-level boards, and project templates. However, challenges can arise when scaling across independent teams or organizational units due to the limited enterprise portfolio views available. Jira excels in terms of scalability. When paired with Align, it provides a wide range of tools for scaling up projects. Microsoft Project is well-suited to its intended purpose. However, this thesis focuses on agile and other software development methodologies. In that case, Microsoft Project does not scale well natively when thinking about developers. Managerial level scaling in software projects works well in more hierarchical settings.

Code collaboration is critical in software development, and its effectiveness was tested through source control integration, code reviews, and CI/CD pipeline con-

nectivity. GitLab excels in this area due to its native integration of version control (Git), merge requests, code review tools, and CI/CD workflows. Jira can be integrated with GitLab, and issues can be linked to merge requests, for example. This offers significant benefits in relation to GitLab. However, is this too much work? Is the automation process or project setup too complex? Jira can also integrate code analysis tools and many other tools that facilitate code collaboration. Microsoft Project does require external tools for all code related activities. Microsoft ecosystem can use Azure DevOps as its platform. Azure DevOps also integrates well with GitHub, since GitHub is owned by Microsoft. This has its advantages. However, is this too much work and configuration? While larger companies are more likely to benefit from the combined use of Microsoft Project, Azure DevOps and GitHub due to their broader resource capacity and complex project portfolios, the benefits for smaller organizations may be less pronounced. For these smaller firms, the time and effort associated with configuring and maintaining such integrations and combinations of tool uses may outweigh the potential advantages.

Regular project management functionalities, such as task tracking, communication, documentation, and reporting, are present to varying degrees across the platforms. GitLab includes basic project management tools such as issue boards, milestones and wikis. It supports Markdown-based documentation, task linking, and comments within issues. Jira supports these features too. Jira also has many other features related to automation, custom fields, and complex workflows. Jira has calendars and can integrate with Slack, for example. These integrations provide flexibility and scope for automation if required.

Findings from this testing indicates that no single tool can comprehensively address all requirements across the multiple dimensions of software development and deployment. Consequently, the selection of tools should be carefully aligned with an organization's methodology, team structure, and technical integration needs.

This observation is reinforced by prior findings and research demonstrating some variability and confusion in both DevOps and Agile practices across different teams. Notably, differences are evident in the adoption and execution of practices. Choice and configuration of supporting tools is also important, underscoring the importance of contextual adaptation in methodology and tool implementation.

4.5 What AI tools and functions there are to help with project management?

In recent years, artificial intelligence (AI) has emerged in many fields, including software development and project management. Integrating AI into project management tools aims to improve project management processes. This section explores the role of AI in project management, focusing particularly on tools such as Jira, GitLab and Microsoft Project. It highlights current features and future developments.

During the creation of the sample project and to-do list application, many AI-like features were identified and used. The AI functions found in Jira during the creation of the sample project include smart issue summarization, automated ticket classification and prioritization, and smart automation suggestions. For example, when creating multiple projects, Jira can detect and analyze past projects. Based on this, it can modify the new template to some extent. AI tools were also used in the creation of the sample project. Some of the automation and issues were primarily created using AI features. According to Das et al.[53], Jira seems to work well with AI-automation.

GitLab on the other hand has CI/CD optimization and risk identification. Perhaps also some other code related features. Microsoft Project doesn't feature many AI-powered plugins. However, it could be used alongside with Microsoft Co-pilot or

Microsoft ecosystems other tools. Microsoft has Excel, PowerPoint, GitHub, Azure DevOps, Word, Power Bi and many other tools which could (and are) theoretically be integrated into Microsoft project. This could bring extraordinary possibilities regarding automation of the project management processes when implemented seamlessly.

Based on shah et als. [54] findings, AI has gained significant popularity among software development environments. It is mostly used in code generation, but also in debugging, testing, project management and deployment. Based on their findings, AI seems to have steep learning curve which requires more resources.

Weng [55] on the other hand argues that generative AI tools such as ChatGPT can enhance software project management by automating repetitive tasks, assisting in planning and scheduling, identifying risks, monitoring progress, and improving communication within distributed teams. These AI powered or integrated tools can provide intelligent summarization, backlog prioritization, and status reporting. These may increase efficiency and situational awareness. However, Weng emphasizes that AI should function as an assistive technology rather than a replacement for human oversight. Wengs main concerns were related to data privacy, model transparency, and the need for new competencies in AI literacy among project managers. AI may allocate tasks to team members. Does it understand small nuances and specific limitations of the specific person? According to Jarrahi[56], AI may lead to more complex projects and more sufficient work which may lead to layoffs. One of the respondents in the survey in chapter 5 also said that they had been laid off due to AI.

However, AI is still probably going to expand in the usage of project management tools. There could be personalized project optimization where AI recommends process improvements and explains why those are a problem. It could provide a lot of automation across tools. This could optimize the workflow of the company.

5 Survey

To better understand how project management tools are utilized in real-world software development environments, a survey was conducted. The objective was to gather insights into tool adaptation, methodology alignment and usage, satisfaction levels, features and emerging trends such as the integration of AI technologies into project management tools and workflows.

5.1 Study design

Due to the purpose of this survey, a decision was made to get participants from around the world and from different companies. Participants came from various backgrounds, including developers, project managers, product owners and executives, representing companies of different sizes. The survey was designed to capture diverse perspectives, from those just at the start their careers to more experienced professionals with multiple years in the industry. The survey was structured to gain more detailed information as well as broader uses of different tools. It combined multiple-choice questions, Likert-scale evaluations (ranging from 1 to 5), and open-ended questions to allow participants to elaborate on their experiences and thoughts. This survey was distributed to an Agile specific discord server "Agile water cooler" and international Facebook groups: "Agile project management", "Agile and scrum framework", "Agile, SAFe, Scrum master" and "Agile project managers". With those the survey got approximately 50-80 answers. In order to gain a few more

participants a decision was made to share it in Finnish Facebook group called " Tekoäly (AI)" and via LinkedIn. The survey gained 110 answers in two weeks total and was conducted in 2025, march. The survey consisted of 19 questions and 4 of them were mandatory.

Respondents were first asked to provide basic demographic information, including their geographical location, the industry their company operates in (allowing multiple selections), their current role within the organization, their years of experience in software development and the size of their organization.

The next section focused on project management methodologies and tools. Respondents listed which methodologies and project management tools they had used. They were also asked to specify the tool they primarily relied on in their current work environment. To assess participants satisfaction in the tools, the survey included scaled questions evaluating the satisfaction with their primary project management tool(s), how well their tools aligned with the methodologies used in their teams or organizations, the importance of tool integrations (such as Jira with GitHub) in their workflow and how well their tools supported remote or hybrid collaboration.

In addition, the survey asked key functional aspects by asking participants which features they found most useful and what challenges they encountered with their current tools. The survey included questions regarding interest in AI-powered features, the perceived impact of AI on project management practices, and the potential for future improvements.

Finally, respondents were asked to provide open-ended feedback, including what they considered to be the most critical feature of a project management tool and suggestions for improvements to current tools or methodologies. This was made to collect qualitative insights regarding tools, AI and many others.

5.2 Results

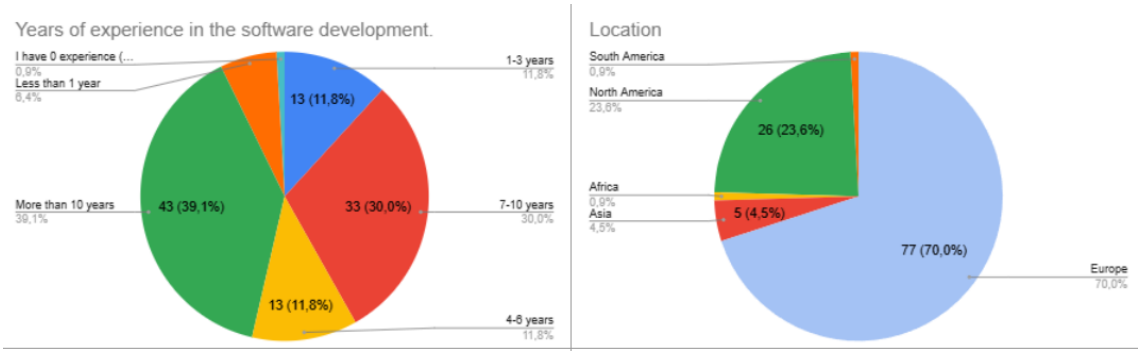


Figure 5.1: Survey participants showing the results of experience in software development and location.

The survey collected responses from participants residing in a different parts of the world. They were primarily based in Europe (77) and North America (26) with additional people from South America (1), Africa (1) and Asia (5) as seen on Figure 5.1.

Participants in the survey represented a wide range of professional experience levels in the field of software development. The largest group had more than 10 years of experience (43, 31.9%), while the second highest group had 7-10 years of experience (33, 30%). Third largest group were tied between 4-6 years of experience (13, 11.8%) and 1-3 years of experience (13, 11.8%). Participants with no experience (1) and with less than one year (7, 6.4%) were minority. The survey was published mainly on Facebook groups and LinkedIn which may give us the answer on why many were seasoned experts since Facebook has more adults as users compared to some other social media platforms.

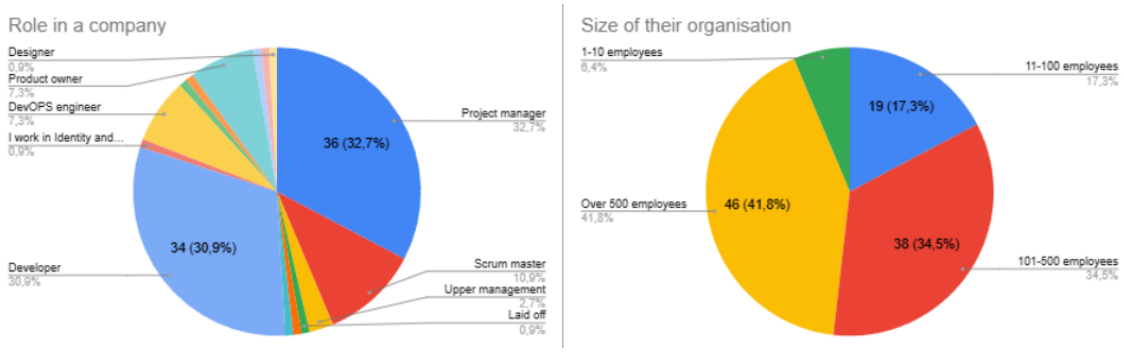


Figure 5.2: Survey respondents' role in a company and size of their organisation

The survey collected data on respondents' current roles within their organizations as illustrated in figure 5.2. The respondents' represented a variety of positions, including software developers, project managers, product owners, team leads, quality assurance specialists, and executives. The most commonly reported role was that of project manager (36, 32.7%), followed by developer (34, 30.9%) and scrum master (12, 10.9%).

In addition to their role, respondents' were asked to indicate the size of their current organization. Respondents were employed in organizations of various sizes, ranging from small companies with 1–10 employees to large companies with over 500 employees. A significant proportion of the responses came from individuals working in companies with 100 or more employees. That was 84 people and 76.3% of the total amount of responses. This could indicate that most participants are not working in start-ups or small companies with only few employees. Therefore tool selection may be a bit different.

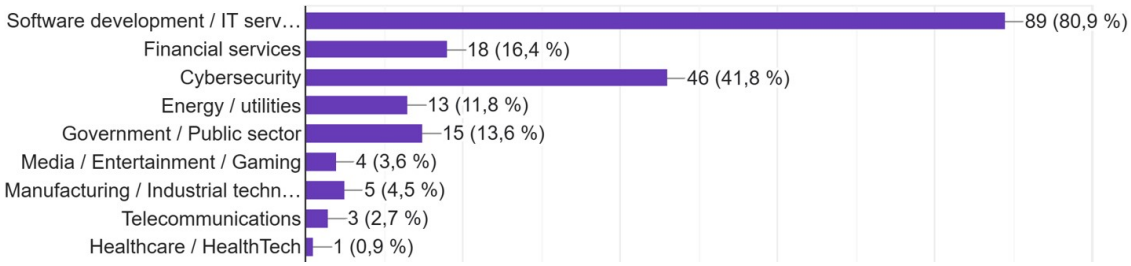


Figure 5.3: Industry Sectors Represented by Participants

The survey collected data on participants' industry that their company currently operates in (figure 5.3). This was made to understand the broader context in which project management tools are utilized. Could there be any relations between different industries and their used project management tools or methodologies? This question was multiple choice answer since there may be companies that operate in multiple sectors. For example company could be providing software solutions as well as providing cybersecurity applications. Most responses were software development or IT-services which concluded 89 people which was 80,9% of the responses as seen in the figure 5.3. The second most common answer was cybersecurity which had 46 responses or 41,8%. 36 of those 46 also answered "software development or IT-services" in the survey. Financial sector had 18 (16,4%) answers, government or public sector had 15(13,6%) answers and energy or utilities had 13(11,8%). Ten respondents also answered 'financial services' and 'software development or IT services'. This could indicate that most people who responded to this survey are in direct contact with software development in some capacity.

IT services and software development firms naturally constitute a large proportion of the sample. However, there are also responses from other industries, indicating that many sectors use software development tools and methodologies to optimize their workflows, whether their work is directly or indirectly related to software development.

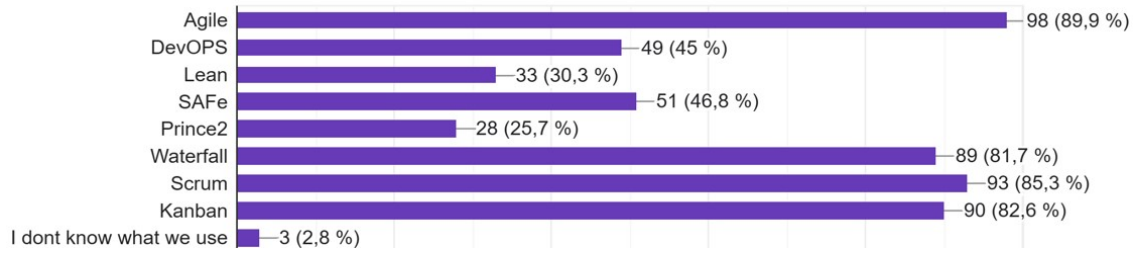


Figure 5.4: Project management methodologies used

One of the main objectives of this survey was to examine the use of project management tools. To understand which tools are being used, it is also important to understand the use of methodologies. Participants were asked to indicate which methodologies they had used in their projects, and they could select multiple approaches to reflect hybrid, overlapping or evolving practices.

The Agile methodology was by far the most dominant as seen on figure 5.4. It was reported by 98 participants (89.9%). This reflects the huge support towards iterative and flexible project environments. Closely aligned with this trend, specific Agile methodologies such as Scrum and Kanban were also widely reported, with 93 (85.3%) and 90 (82.6%) participants. These findings support the notion that Agile principles are deeply embedded in software development projects and widely used.

Interestingly, the Waterfall methodology also received a high response rate, with 89 participants (81.7%) noting its use. This suggests that while Agile methodologies dominate in the field, traditional approaches still maintain a strong presence. This may be due to legacy systems, project-specific requirements or sector-specific demands. Some sectors may be less conservative regarding managing projects and people which could lead to waterfall being used so often. This may also mean that agile and other methodologies are often paired with more hierarchical methodologies such as waterfall. There could be a hybrid way to manage companies where higher management is using more hierarchical way of operating and developers and teams

are using the flexible way. This was shown in the previous chapter as some tools support both Waterfall methodologies as well as Agile. Also the nature of Microsoft Project supports waterfall methodologies.

Other notable methodologies include SAFe, selected by 51 participants (46.8%) and DevOps with 49 responses (45%). These approaches reflect the interest and use cases in scalability, automation and tighter integration between development and operations. Lean (30.3%) and Prince2 (25.7%) were less commonly used but still demonstrate the different methodologies used by participants'. The presence of Lean principles indicates attention to process efficiency and added value due to processes. Participants' that answered lean were notably in government, public sector or in energy sector. Energy sector could be argued that it is somewhat similar to manufacturing. Government and public sector had 15 participants where 13 of them used lean. Agile was used by 12 of them and waterfall 13 of them.

Notably, a very small percentage (2.8%) of respondents indicated that they did not know which methodology their team or company used. This means that the majority of participants are actively engaged with or at least aware of the management approaches in their companies.

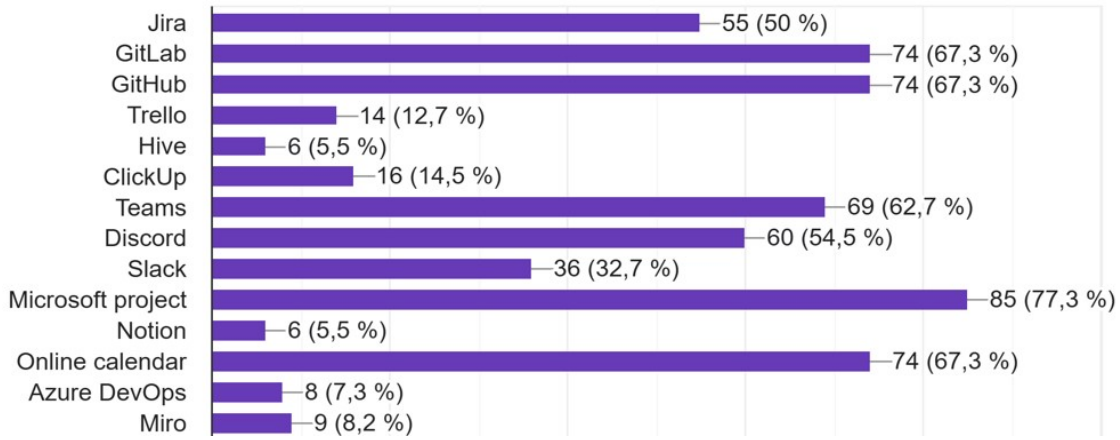


Figure 5.5: Project management tools used

Understanding which project management tools are currently in use across software development teams was a key objective of this survey. Participants were asked to indicate all the tools they had used in their software development projects. This helps us understand different tool ecosystems, hybrid tool uses and combinations of tools used.

The most widely used tool among respondents was Microsoft Project (figure 5.5), selected by 85 participants (77.3%). Interestingly, 75 out of 84 people in companies with over 100 employees used Microsoft Project. Microsoft project was also used heavily when people were using Waterfall methodologies and Lean principles. This finding may reflect the tool's established use cases in many enterprise environments and large companies. Microsoft Project's strong showing suggests continued use on traditional scheduling and resource management particularly in large companies.

Second most used tools were GitHub, GitLab and Online Calendars, each selected by 74 participants (67.3%). The use of Git-based platforms (GitHub and GitLab) indicates the close relationship between project management, version control and code collaboration. These platforms not only facilitate source code management but also support regular project management needs. Git-based platforms were used mostly by developers and agile specific roles such as scrum master based on the survey. Online calendars were used by wide variety of people with no significant findings.

Microsoft Teams (62.7%) and Discord (54.5%) were also heavily represented, telling the importance of communication tools as components of project management. While they are not traditional project management tools they are still widely used especially in remote or hybrid teams. Interestingly, Teams was not used more. This could mean that participants' didn't understand that it was Microsoft teams or that they prioritized the tools. Teams was used more on the larger companies where as discord was used more with smaller companies based on survey findings.

Other tools such as Jira (50%) and Slack (32.7%) further reinforce the trend of agile methodologies. Jira, associated with Agile teams and issue tracking is somewhat used tool. Jira also offers integration with other tools such as GitHub. Almost all developers that used Jira also used GitHub or GitLab. Slack, though primarily a communication platform, is frequently integrated with other tools to support project transparency and agile coordination. This tool was also related to developers.

Less widely used tools were ClickUp (14.5%), Trello (12.7%), Azure DevOps (7.3%), Miro (8.2%), Hive (5.5%), and Notion (5.5%). Some of these tools may be more relevant to niche teams, whereas Jira is used more widely. 23 out of 26 people working in companies with fewer than 100 employees use one of these tools or Jira. Some of these tools have free versions, which could encourage small companies to use them, even though most require payment for commercial use. The small use of Azure DevOps seems to be interesting since it is a Microsoft owned software.

These findings reveal several important dynamics. Firstly, there is no single dominant project management tool. Teams tend to use multiple tools, often combining communication platforms, task tracking, version control, and traditional project management software. Integrating tools (e.g. GitHub with Teams or Slack) appears central to many workflows. This suggests that tool integration is a viable strategy for using these tools. This is because there may not be a tool that can do it all.

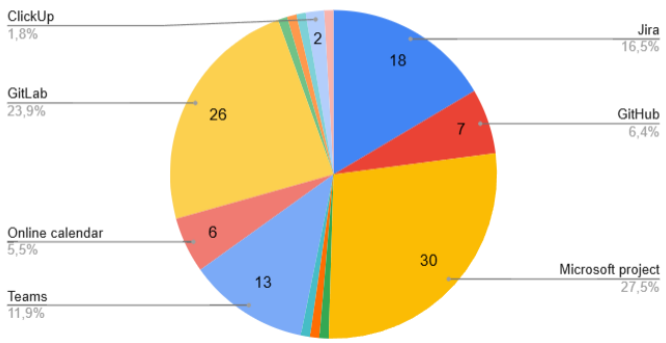


Figure 5.6: Primary project management tool in use

To further understand tool usage, the survey also asked participants to indicate which project management tool they considered their primary tool they rely on most in their day-to-day workflow (figure 5.6). This question is aimed to ask whether regular project management tools, communication tools or code collaboration tools are the most important ones in daily life of software development.

The most commonly cited primary tool was Microsoft Project, selected by 30 participants (27.5%). Microsoft project was the primary tool of 25 out of 36 project managers. Almost no developer said that this is their primary tool. This is consistent with earlier findings regarding overall usage and reaffirms Microsoft Project's entrenched role in many organizations, particularly in large companies. Its prominence most likely reflects both organizational preference and its integration into the broader Microsoft ecosystem. Since many companies use other Microsoft products.

Following Microsoft Project, GitLab was the second most used primary tool, named by 26 participants (23.9%). Github got 7 answers (6.4%). This result highlights the growing significance of platforms that merge source control with other project management tool features. Jira accounted for 18 responses (16.5%) probably within smaller Agile and Scrum teams. Jira was a primary tool for people that are in small to large sized companies.

Teams (11.9%) and Online Calendars (5.5%) were also identified as primary tools by some participants'. Interestingly, the presence of communication tools like Teams and calendar-based systems as primary project management tools may suggest that in some teams coordination and scheduling are prioritized over comprehensive task tracking or process enforcement. Or the calendars could be used for task tracking. Other tools such as ClickUp (1.8%) appeared only minimally as a primary tool.

These results offer several key insights. While many organizations use multiple tools, they often gravitate towards one or two central platforms for core project planning. The differences in primary tools reflect the different types of organiza-

tions, with some prioritizing more traditional tools (e.g. Microsoft Project), some focusing on developer-centric ecosystems (e.g. GitLab), and some relying on communication and scheduling platforms. Based on this survey, the size of the company and role in the company could be the biggest factors in determining which tool is used primarily for project management. This may be because larger companies tend to have multiple teams, for which Microsoft Project may be more useful. Smaller companies on the other hand could just use one tool that does it all, such as GitLab where scaling of project management processes may not be so important. Also if a Developer works alone on code related tasks (e.g analytics), there may not always be need for a Git-based system. In that case Developers main tool could be Jira or some communication tools, for example.

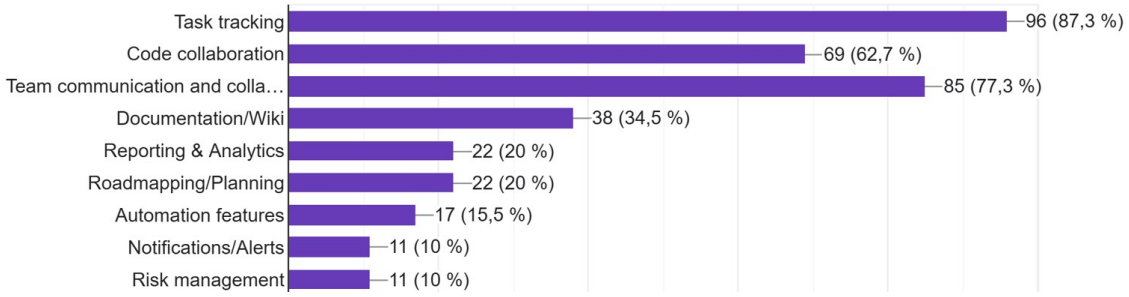


Figure 5.7: Most valued features in project management tools

To better understand why employees would use specific project management tools, the survey asked about the most valued features. This question was designed to reveal why someone would use a particular tool or combination of tools. Could this explain why some tools are superior to others?

The most frequently answered feature was task tracking (figure 5.7), with 96 responses (87.3%) indicating it as a critical component of their project management toolkit. This finding aligns with other findings in the survey like what tools are being used as primary tools. Microsoft project, Jira, GitLab and many others all

offer some sort of task tracking.

The second favorite was team communication and collaboration, favored by 85 participants (77.3%). This is also in line with the primary tools and tools overall. The rise of hybrid or remote work could also be a factor in this since many software development employees may do either hybrid or remote work. Code collaboration was the third most valued feature, selected by 69 respondents (62.7%). Most developers answered this which is understandable. In larger coding projects version control and code collaboration is important. Interestingly, many project managers answered this as well.

Documentation and wiki support was answered by 38 participants (34.5%). Knowledge management and version control could be a factor for this. In larger projects, it is vital to pass on the information of how everything works. Reporting and analytics, road-mapping and planning were both highlighted by 22 respondents (20%).

Lesser-valued features included automation tools (17 responses, 15.5%), notifications/alerts and risk management (both with 11 responses, 10%). These results may indicate either under-utilization of these features, or that employees from a wider range of backgrounds perceive them as less important. This could suggest that some people find these features useful, while others do not use them at all. In contrast, communication of some sort is used by almost every employee.

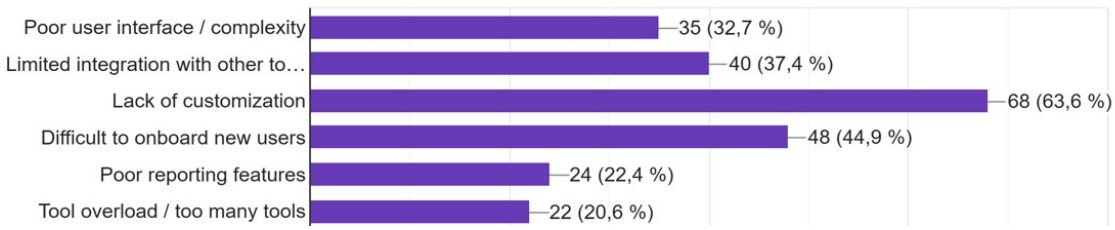


Figure 5.8: Challenges encountered with current project management tools

In addition to assessing favored features, the survey also asked about common challenges users face when employing project management tools in software development contexts. The most frequently reported challenge was a lack of customization (figure 5.8), answered by 68 respondents (63.6%). This highlights a significant gap between user needs and tool flexibility. This suggests that many tools fail to accommodate diverse workflows, team structures or project types. This was also mentioned in many open ended questions. Some also criticized that they are being specifically made to support SAFe for example. One developer wrote: "They shouldn't be built to support SAFe. It makes it hierarchical and too complex". Which could indicate that the customization may be done but it gets limited by some other features or it becomes too complex if the employee doesn't need those features.

Another prominent issue was the difficulty of onboarding new users, reported by 48 participants (44.9%). This points to usability and learning curve concerns. This can make tool adoption more complex and reduce overall team productivity.

Limited integration with other tools was a challenge for 40 respondents (37.4%). In agile development seamless integration with version control systems, CI/CD pipelines, communication platforms and other software is critical. This challenge was mainly reported by people that used Microsoft Project, Teams or Discord. Only 8 out of 55 Jira users reported this as a challenge or a problem.

A poor user interface or excessive complexity was mentioned by 35 participants (32.7%). Less common but still notable challenges included poor reporting features (24 responses, 22.4%) and tool overload or the perception of having to manage too many separate tools (22 responses, 20.6%). These issues suggest that users may feel overwhelmed by the number of specialized tools in use, and that they may not find the analytics features sufficient for monitoring project health. One developer left an open comment regarding tools: 'As a developer, I haven't really liked any of the tools I have used. I am lazy, so if tools are difficult to find or update, or do not fit

my logic, I do not want to use them. If many people don't do it, the tool is not worth using." This response suggests that some of the tools are ineffective or the configuration may not be suitable for specific needs.

These findings highlight a tension between functionality and usability. Although project management tools offer many features, they are undervalued if they are inflexible, difficult to learn or poorly integrated. It is important to understand which tools are necessary and which are more of a bonus. Otherwise people may become frustrated.

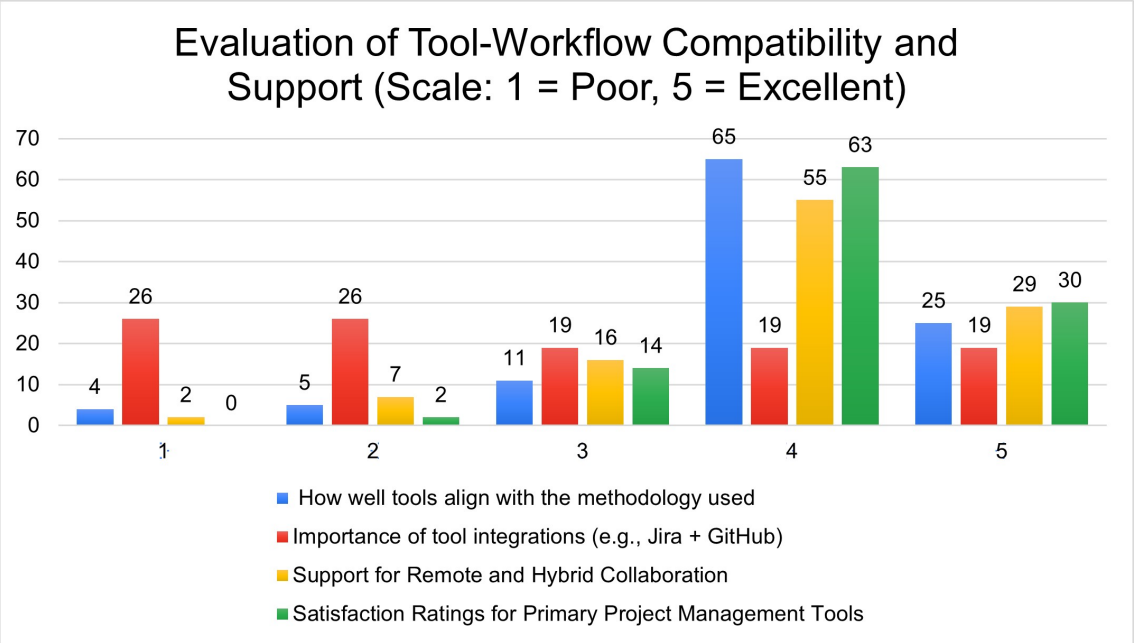


Figure 5.9: Evaluation of tool-workflow compatibility and support (Scale: 1 = Poor, 5 = Excellent)

To assess perceptions of tool effectiveness in software development environments, survey participants were asked to rate four aspects of their project management tools using a five-point Likert scale (1 = Poor, 5 = Excellent). The four questions evaluated were: (1) alignment with development methodology, (2) importance and performance of tool integrations, (3) support for remote and hybrid collaboration,

and (4) overall satisfaction with primary tools.

A substantial majority of participants rated their tools positively in terms of alignment with their project management methodology (figure 5.9). Specifically, 65 participants rated this as a 4, and 25 rated it as a 5. This suggests strong perceived compatibility with Agile, Scrum or other approaches. Low ratings were rare only 4 for 1 and 5 for 2. This indicates a generally high satisfaction for the tools used.

Integration with other tools (e.g. Jira and GitHub; CI/CD pipelines) showed some variation. While many users acknowledged its importance, 26 participants gave it a rating of 1 or 2, 19 gave it a rating of 5 and another 19 gave it a rating of 4, suggesting that many people do not use tool integrations as they are not part of their workflow. Developers and GitLab or GitHub users rated this higher than project managers or scrum masters did.

Support for remote and hybrid work received positive feedback. The majority of participants gave high marks: 55 rated it as 4 and 29 as 5, indicating that these tools are relatively useful in different work environments. Whether they are hybrid, remote or on-site.

The general opinion was that the primary tools used were useful. A total of 63 participants rated satisfaction at 4, while 30 gave a 5. This indicates that most teams are relatively satisfied with their current solutions. Very few respondents rated their tools poorly.

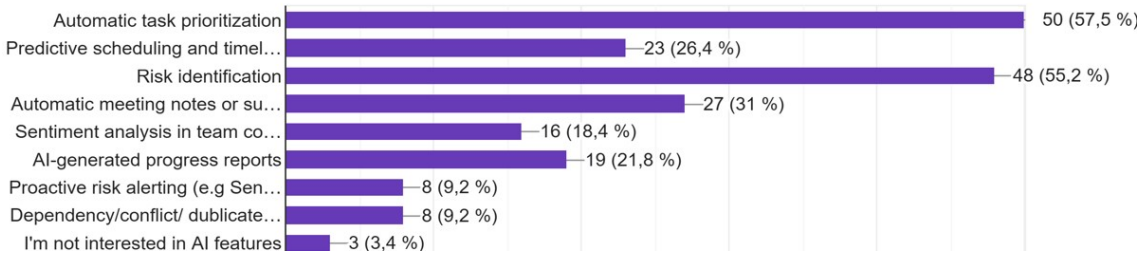


Figure 5.10: Interest in AI-powered features in project management tools

With the rapid integration of artificial intelligence (AI), it could be important to figure out what AI-powered features would be most useful. The question explored which AI features participants would find most valuable. This question was answered by 87 people out of all 110 as seen on figure 5.10. The most desired AI feature was automatic task prioritization, selected by 50 respondents (57.5%). Close behind was risk identification, with 48 responses (55.2%).

Other notable preferences included automatic meeting notes or summaries (27 responses, 31%). This could be a feature that could streamline documentation. Project managers found this the most useful. Predictive scheduling and timeline estimation was also valued, with 23 responses (26.4%).

Interest in AI-generated progress reports (19 responses, 21.8%) and sentiment analysis in team communication (16 responses, 18.4%) suggests that automating some processes in projects is needed. Less commonly selected features included proactive risk alerts and dependency/conflict/duplicate ticked detection, both with 8 responses (9.2%).

The survey had also 4 open ended questions: (1) Has the rise of AI impacted how project management tools are being used in day to day life?; (2) In your opinion, what is the most critical feature a project management tool should have? ; (3) What would you change or improve in the tools or methodologies you currently use? ; (4) Open answer slot regarding Project management tools in software development.

These questions got 42 written answers overall, offering valuable insights that complements and explains some of the quantitative findings giving qualitative insights. The responses were thematically analyzed to identify recurring themes and suggestions.

For the first question many people answered that the tools have not yet changed much due to AI. 9 out of 14 people said that it has not changed or the change is insignificant. One person said that they and their team got laid of due to AI. 4 out

of 14 people reported that AI has changed project management tools in some way. One developer commented: "It impacts on the day to day activities, I am relying more and more on these tools and features".

The responses for the second question were not aligned with each other. All 13 of the answers were somewhat different. There were responses such as "ease of use", "clarity of role and employees tasks, easy to navigate and use UI", "Producing project views for executives", "thorough internal manual" and "An easy way to follow progress". These show that everyone's needs are different and their views vary depending on their role, project and companies. However many showed interest in the ease of use overall.

Responses to the third question frequently addressed complexity, poor user experience and limited flexibility in current tools. Suggestions ranged from simplifying interfaces to improving cross-platform integration and aligning better with specific development methodologies. One scrum master wrote: "I'd enhance AI-driven features to automate bug triaging, sprint planning and code review". Whether it is the company product or their tools - in agile everything should be flexible and continuously improving.

The fourth, open-ended prompt allowed respondents to express broader opinions. This question got 5 answers where 4 of them criticized the tools overall. They were not satisfied with the tools or the amount of tools used. One developer expressed that "we should just focus on the final product not the trying to over dramatize the need of project management tools".

In conclusion, the survey provides deeper insights into patterns of tool usage. It offers support for the previous findings that both tools and methodologies are highly context dependent. The findings reaffirm earlier research showing that organizations and teams interpret, adopt, and apply these tools and methodologies in diverse ways, shaped by their specific operational, cultural, and technical environments.

6 Discussion

This thesis explored the role and effectiveness of project management tools, such as GitLab, Microsoft Project and Jira. This chapter takes a closer look at the findings from the fourth chapter and the survey. Chapter 4 included a basic summary of the findings.

6.1 Summary of key findings

Empirical testing and analysis in this thesis of the nuances of project management tools used in software development has provided valuable insights into their methodological alignment, scalability, and practical utility. GitLab, Jira and Microsoft Project all have their own strengths and limitations. They are each designed based on their intended use cases and underlying design philosophies. These tools have evolved significantly over time, and will continue to do so in the future.

Based on the sample project, GitLab has emerged as a highly effective solution for Agile and DevOps-driven environments. Its integrated approach combines version control, continuous integration and delivery (CI/CD), and issue tracking to provide development teams with a seamless workflow. The platform's native support for merge requests, code reviews, and automated pipelines aligns well with iterative development cycles. This makes it particularly advantageous for small to mid-sized Agile teams. However, GitLab's scalability in large-scale enterprise settings is somewhat limited.

Jira has demonstrated flexibility in accommodating various project management methodologies. It offers great support for Scrum and Kanban, and to some degree SAFe. Its ability to integrate with external version control systems (e.g. GitHub and GitLab) and CI/CD tools enhances its utility in DevOps contexts. Jira also integrates well with many other tools, such as Figma, Slack and Teams. Advanced features such as customisable workflows, cross-team dashboards and Jira Query Language (JQL) provide control over project tracking. This makes Jira a good option for scaling teams and organizations. However, Jira's complexity and steep learning curve can present challenges for new users. Since it is a supporting process tool not everyone wants to spend time mastering it. Accessing its full potential often requires premium-tier features or other Atlassian products, such as Jira Align. Overall, Jira seems an extremely promising tool, or better yet, an integration platform. This gives it huge advantages in terms of automation and the seamless integration of project management processes. Another advantage of Jira is the visualization of everything. There are multiple visual and text-based ways to follow and manage projects.

Microsoft Project is excellent for traditional, plan-driven project management. Its strengths lie in detailed scheduling, resource allocation, and Gantt-chart visualizations, all of which are relevant to the Waterfall methodology. However, its rigid structure and lack of native support for Agile or DevOps workflows restrict its use in modern software development especially within developers. While it can be adapted to Agile frameworks through manual or external configurations, the effort often outweighs the benefits. This is especially the case when compared to tools designed specifically for Agile and DevOps, such as Jira or GitLab. Nevertheless, Microsoft offers similar tools, such as Project for the Web, Azure DevOps and GitHub, as well as cloud-based solutions. Microsoft also offers significant integration and automation possibilities thanks to its large ecosystem of tools. Therefore the differences between Jira and Microsoft Project may be smaller than expected. Microsoft's large arsenal

of native tools gives it potential automation advantages in the future.

Next, we will take a closer look at the survey results. This thesis explores the use, preferences and challenges related to project management tools and methodologies within software development environments through a survey. A total of 110 participants responded, representing a variety of locations, industries, organizational sizes, and roles. The primary aim was to understand how professionals manage projects and which tools and methodologies they use, as well as which features and improvements they value.

Most respondents were based in Europe (77) and North America (26), with a majority having significant professional experience in software development. Nearly 70% had more than seven years of experience with 43 reporting over a decade in the field. Participants held a range of professional roles, most commonly project managers (36), developers (34) and scrum masters (12). Organizational sizes were toward larger firms: over 76% worked in companies with more than 100 employees.

Respondents primarily operated in software development or IT services (80.9%), with a considerable presence in cybersecurity (41.8%), finance, government and energy. The survey revealed widespread use of Agile methodologies, with Agile in general (89.9%), Scrum (85.3%) and Kanban (82.6%) dominating. Notably, Waterfall was still in use by 81.7% of respondents, suggesting the need of hybrid project management approaches. This could also mean that operational level operates on agile methodologies and managerial level operates on more traditional ways. Methodologies like SAFe (46.8%) and DevOps (45%) also had significant representation, while Lean and Prince2 were used more selectively, particularly in the public and energy sectors.

Microsoft Project emerged as the most commonly used project management tool (77.3%), especially among project managers and large organizations. GitHub, GitLab and online calendars (each used by 67.3%) highlighted the intersection of version

control, collaboration, and scheduling. Tools like Jira (50%) and Slack (32.7%) reflected strong Agile support, while ClickUp, Trello, Miro and Notion were more used in smaller companies. A multi-tool approach was common, with teams combining communication platforms, scheduling tools and code repositories.

When asked about their primary tool, Microsoft Project (27.5%) was most cited especially by project managers in large organizations. In contrast, GitLab (23.9%) was more prominent among developers and smaller organizations. Jira (16.5%) was used across the board from small companies to large companies. This reflects my personal real world findings that Jira is used widely in Finland. Microsoft Teams and online calendars also served as primary tools for some which indicates variation in workflow.

The most valued project management features were task tracking (87.3%), team communication (77.3%) and code collaboration (62.7%). Major challenges included lack of customization (63.6%), onboarding difficulty (44.9%), poor integration (37.4%) and complex interfaces (32.7%). Respondents emphasized a desire for simplicity, flexibility and integration, especially in environments where multiple tools are used. Lack of customization and high use of Microsoft project may correlate each other. 3rd party integration's are not well established in Microsoft Project where as Jira has a huge selection of 3rd party tools. Even the developer could somewhat easily create their own plugins to help automate Jira.

Participants generally reported high satisfaction with their tools, especially in how they align with methodologies used (82 rated it 4 or 5 out of 5) and supporting remote work. However, integration quality received more mixed feedback. AI-powered features were of strong interest. The most desirable future AI features included automatic task prioritization (57.5%), risk identification (55.2%) and automated meeting summaries (31%).

Qualitative, open-ended responses revealed skepticism towards tool overload and

complexity. While some respondents viewed project management tools as essential, others expressed frustration with tools that hinder productivity. AI adoption was considered limited but promising, with some already experiencing the benefits of automation and planning.

Comparing the sample project with the survey results reveals clear similarities and differences in how project management tools are perceived and used. In both the simulated sample project and the survey responses, it was necessary to use multiple tools to cover the different aspects of project management. Microsoft Project offered strong planning capabilities but was considered extremely limited. Similarly, the complexity of Jira was acknowledged in both, and it was valued for its flexibility, but often criticized for its steep learning curve and the effort required for configuration.

GitLab seemed ideal for light Agile workflows and closely aligned with the preference of survey respondents for integration with development environments, offering straightforward task tracking without complicating background tasks unnecessarily. However, the lack of planning features highlighted in the sample project was not as prominently flagged in the survey. This possibly indicates different expectations and needs for such tools among users who favor them for simplicity. This comparison confirms that the choice of tool is strongly contextual, depending on team needs, workflow structure, and project complexity. Both data sources affirm that no single tool can comprehensively meet all requirements.

Both the sample project and survey data confirm widespread hybridization of methodologies and tools. In the sample project, the three tools were tested across a spectrum of Waterfall and Agile practices, revealing strengths and limitations in each context. Survey results support this, with more than 80% of respondents using Agile, Scrum, Kanban and Waterfall simultaneously. This underscores the coexistence of multiple methodologies within the same organizations or even the same teams. Methodologies may not be so polarized when choosing a methodology

for a company or organization. Mix of methodologies could be used or partial usage of methodologies could be used according to the survey. Perhaps this is the reason why some of these methodologies can be seen as ideologies.

It is clear that flexibility and configurability of tools are critical for accommodating hybrid processes and workflows. Jira was found to support this well, whereas Microsoft Project imposed a more rigid structure. The survey results reflected this trade-off: while some users valued structured, simplified tools, many were frustrated by tools that lacked adaptability to their working practices. However, based on survey results, Jira was used significantly less than Microsoft Project. Perhaps separate tools are better than highly configurable and maintainable tools in some cases.

Both the hands-on evaluation and the survey highlighted similar limitations. Complexity, poor customization and integration issues emerged as recurring challenges. The sample project revealed that, even with powerful tools such as Jira and Microsoft Project, significant effort is required to adapt them to actual workflows. This concern was mirrored by survey respondents.

Interestingly, while the sample project revealed operational inefficiencies when switching between tools or compensating for missing features, the survey revealed that users became frustrated and resistant when the tools failed to meet their needs or caused them to duplicate their efforts. This suggests that, in addition to technical capability, usability and team alignment are also important factors in tool adoption. It is not about perfecting the tools and optimizing processes, but also about the human side of it all. Humans are not robots and they may at times require imperfections for optimized workflows.

AI features remained mostly underutilized in both the sample project and the survey. The sample project encountered few built-in AI tools, making their usefulness difficult to assess. However, the survey revealed a high level of interest in predictive or automated features, particularly among developers and operational staff.

This indicates an unmet demand for intelligent, context-aware project support tools, although their actual usefulness remains to be seen. Jira had automated scheduling tools that would reschedule tasks based on previous progress. This feature worked, but its usefulness could not be determined.

6.2 Comparison with previous research

The findings of this thesis can be effectively contrasted with the earlier work of Brad et al. (2016)[57], which examined the impact of project management tools on IT project success. It identified integration and usability as important success factors which goes in line with this thesis as well. It highlighted tools that enhance team communication and visibility, aligning with this thesis' findings on the value of integrated ecosystems. The paper also compared Jira with Microsoft team foundational server nowadays known as Azure DevOps. According to the paper, Jira had more features and compatibility compared to Microsoft team foundational server at the time of the paper (2016). This indicates that Jira may be in-between Microsoft Project and Azure DevOps functionally. This raises questions. Is it better to have one managerial tool focused on Agile and one managerial tool focused on traditional methods, rather than combining those into one tool? The survey conducted for this thesis may provide an answer: Microsoft Project was used more widely than Jira. However, Azure DevOps had very few users.

A survey conducted in 2011 by Azizyan et al. [58] was based on 121 responses. Survey asked about the most commonly used project management tool. Physical wall (26%), Microsoft Project (8%), Rally (5%), Mingle (3%), VersionOne (2%), Jira (2%) and Team Foundation Server (2%) were the tools in the survey. This highlights the fact that toolset has changed significantly in the past years.

A separate survey conducted in 2013 provides an alternative perspective on the adoption and satisfaction levels associated with development and productivity tools,

albeit not limited specifically to project management software [59]. The survey having 3,501 respondents investigated the usage of various tools and methodologies in software development. The most commonly used tools reported were: Microsoft Excel (66%), Microsoft Project (48%), VersionOne (41%), Atlassian Jira (36%), Microsoft TFS (26%), IBM ClearCase (10%) and LeanKit (5%). In addition to usage statistics, the survey also assessed user satisfaction. The highest satisfaction ratings were recorded for VersionOne (93%), followed by Atlassian Jira (87%), LeanKit (84%), TargetProcess (83%), Microsoft TFS (79%) and ThoughtWorks Mingle (69%). However, this survey was related to VersionOne which may be the reason for specific results.

These findings offer meaningful comparative insights for this thesis. Despite changes in the software landscape over the past decade, notable parallels can be seen. For instance, Microsoft Project continues to be more widely used than Jira, consistent with trends observed in both this thesis and the earlier work by Azizyan et al. [58] and the 2013 survey[59]. Interestingly, the 2013 [59] survey indicates that while Agile methodologies were present, they had not yet reached the level of prominence seen today. Respondents were asked on the categories of overall tools employed in their workflows, with bug trackers (83%), task boards (81%), automated build tools (69%), Agile project management tools (66%) and Kanban boards (43%) ranking as the most commonly utilized. These data points can be also contrasted by the findings of this thesis. One of the most notable difference is bug tracking.

Interestingly, increase in the adoption of Agile project management tools seems to be a growing trend. The 2013 survey [59] reports a 6% year-over-year growth in use between 2012 and 2013, highlighting a clear trend toward Agile adoption. This historical data underscores the momentum Agile practices have gained in the years since. This is further evidenced by the results presented in this thesis.

6.3 Answers to research questions

In this section research questions are answered based on previous chapters.

RQ1: What are the necessary tools in software project management?

The findings from this thesis strongly support the hypothesis that different types of software projects necessitate different project management tools. This differentiation is influenced by several interrelated factors: project management methodology, human preference, team size, organizational size, industry domain and the end product. The selection of tools is thus a strategic decision, embedded in broader organizational logic and process design.

Agile and DevOps centric projects typically require tools that support rapid iteration, collaborative coding and automated testing. For such projects tools like GitLab and Jira are well-suited. These tools offer high-frequency feedback loops, task visualization (e.g., Kanban boards) and integrations with development environments. The emphasis in these projects is on adaptability, team autonomy and reduced overhead. All of these are poorly supported by traditional project planning tools.

In contrast, projects that follow a Waterfall or other traditional methodologies demand a different set of capabilities. Here, Microsoft Project is more applicable. These projects often make flexibility less important than clarity and structure.

Hybrid planning was shown particularly prominent in the survey results. Some projects blend Agile practices at the operational level with traditional planning at the strategic level. For instance, a development team may use GitLab or GitHub and Jira or Azure DevOps for coding, sprint tracking and backlog management, while senior project managers use Microsoft Project for milestone forecasting and budget control.

Team and organization size also play critical roles. Smaller teams with flat hierarchies often prioritize ease of use and minimal configuration, leading them toward

tools like GitLab or Trello. Larger enterprises, on the other hand, require tools that can scale with complexity. This often justifies the overhead of tools like Jira or Microsoft Project, even if adoption requires significant onboarding, maintenance and training.

There are also other tools that are needed in software project management. Communication, documentation, CI/CD, code analysis and other tools are also needed. Some tools fix a smaller problem and some tools work as an integration platform where all the tools combine.

Ultimately, this thesis demonstrates that tools are not interchangeable. They are tightly bound to the methodological, structural and cultural conditions of the projects they support. As the software industry continues to evolve, tool selection will become increasingly strategic. It requires not only technical evaluation but also organizational analysis to ensure alignment with long-term delivery models and team dynamics. As Ailasmaa shows in his findings [42]: When switching tools there are always consequences and processes that need to be changed. Whether it is for the good or for the bad - often both.

RQ2: What benefits does each tool bring to the team?

Each of the three project management tools tested: GitLab, Jira and Microsoft Project offers distinct advantages that correspond to particular types of teams, workflows and organizational goals. The benefit of a tool is not simply a function of its features, but of how well it supports a team's processes, decision-making structures and coordination needs. In this sense, tool "benefit" should be understood as relational: what works well in one environment may cause problems in productivity in another environment.

GitLab provides the most value to teams that prioritize integrated development environments and automation. Its mostly contributes to teams using DevOps pipelines by combining version control, CI/CD and issue tracking into a unified plat-

form. The benefit of GitLab is efficiency through convergence of tools and functions: it reduces operational friction by eliminating the need to jump between tools. It also enhances traceability as every code change is associated with an issue, merge request and pipeline status.

Jira's main advantage is its configurability and methodological flexibility. For teams working within Agile or hybrid frameworks, Jira facilitates nuanced workflow design. The true benefit of Jira is its capacity to serve as a process governance tool: it enables coordination across multiple teams and layers of abstraction (e.g., epics, sprints, initiatives) which is critical for scaled Agile environments. Jira integration capabilities are of significant value as everything is under one platform - or could be controlled under one. This is beneficial to the managerial team. Therefore, Jira could be a useful tool when organization needs and team operations are wanted under one software.

Microsoft Project brings benefits primarily in the form of planning precision and structured oversight. Its strength lies in giving project managers and managerial teams significant oversight by utilizing multiple features. However, it doesn't give much value to the development teams practicing agile or other similar methodologies.

However, a critical insight from this thesis is that benefits are often accompanied by trade-offs. GitLab's simplicity comes at the expense of enterprise-scale planning. Jira's flexibility introduces configuration complexity. Microsoft Project's structure can inhibit Agile compatibility and developer usability. Therefore, the most substantial benefit of any tool may be its ability to align with a team's operational style and rhythm supporting productivity without imposing unnecessary constraints. Could this lead to the realization that hybrid tool usage is absolute necessary since Microsoft, Atlassian or other companies will never produce tools that support all project environments?

In summary, tools contribute to project success not only through their func-

tionalities but through their alignment with how teams think, plan and execute. Understanding tool benefit requires looking beyond technical capability to the social and procedural context in which ways the tool(s) is/are being employed.

RQ3: Is there a universal project management tool in software development?

This thesis confirms that no single project management tool is universally optimal across all software development contexts. Rather than seeking an universal solution, software teams must evaluate tools based on contextual appropriateness. This varies in terms of development methodology, organizational scale, team composition and technical environment. The notion of a "necessary" tool is dependent upon a project's specific requirements rather than determined by specific tool superiority or its features.

The observed differences between GitLab, Jira and Microsoft Project illustrate this contextual dependency. Each tool has been constructed with different assumptions and use cases in mind. GitLab is designed around the principles of continuous integration and Agile workflows while prioritizing lightweight task management and streamlined development pipelines. In contrast, Microsoft Project reflects a more traditional project management focusing on top-down planning, resource allocation and progress forecasting aligned with Waterfall methodologies. Jira occupies a hybrid space, offering a highly customizable environment that can flexibly support both Agile and scaled Agile frameworks often with increased complexity. These findings are also supported by the survey in this thesis. Smaller companies may use GitLab but larger companies need more powerful tools such as Microsoft Project or Jira alongside Git-based environments.

The diversity of tool preferences identified in the survey further supports this view. Organizations were shown to adopt multiple tools in parallel, often integrating version control systems, communication platforms and scheduling tools to build a

specific toolchain. This reflects the industry's broader need for tools with different functions.

From a theoretical perspective, this finding aligns with socio-technical systems theory (STS). It tells that technology must co-evolve with organizational structures, practices and human mind in the mix [60]. A universal tool would imply a universal workflow or organizational structure which does not exist in modern software development. It might even be absolutely impossible and unrealistic - not even an utopistic view. Instead, each tool serves as an artifact embedded in a broader ecosystem of processes, workflows and norms. Organizations must assess tools in terms of their fit with existing workflows and their adaptability to future needs. But still, could there be room for integration platform that Atlassian Jira or Microsoft tries to implement? Could all of these software pieces be integrated into one platform that can control all functions and features under one software? In theory this could make automation easier and simplify the project management itself.

In conclusion, the pursuit of a universal project management tool may be misguided. Tool choice should be reframed as a strategic decision involving trade-offs among flexibility, scalability, integration capacity and ease of use. This thesis and other research affirms that the most effective project management environments are those where tools are tailored to the operational logic of the team and the complexity of the project. This is supported by Brad et al.[57] findings as well.

6.4 Future insights

As the complexity and pace of software development continues to increase, the tools used to manage projects must also evolve. This thesis has examined the current landscape of project management tools, focusing specifically on GitLab, Jira and Microsoft Project. The findings from this thesis, together with trends observed in other surveys, suggest that the future of project management in software devel-

opment is likely to be shaped more by the integration of these tools into broader intelligent ecosystems than by incremental improvements or the addition of new standalone functionalities. However, Microsoft's ecosystem and Atlassian ecosystem differ slightly in how they address this issue. Microsoft does not rely on a single central tool; rather, it provides a set of tools, some of which can be integrated with one another to support project management activities.

The trend towards hybrid workflows, integrations, and multi-tool ecosystems is likely to accelerate since hybrid work and wide range of projects may necessitate those. Organizations are increasingly adopting toolchains rather than standalone tools, combining the best features from different platforms based on the survey and other related surveys shown in this thesis. Artificial intelligence (AI) and machine learning (ML) are probably going to play a role in the future of project management and software development. AI has significant potential for intelligent project tracking, automation and analysis, and for the automated allocation of tasks, for example. Future tools could have more intuitive interfaces, embedded best-practice templates, and automated reporting. Could Git-based environments such as GitLab and GitHub expand their project management features to further appeal to developer teams? Those tools could also give more functions regarding scalability.

Agile, DevOps and hybrid methodologies will continue to evolve. Project management tools will need to support flexibility not only in methodology (Scrum, Kanban, Waterfall, SAFe), but also in working styles. During the past years hybrid and remote work has significantly risen. What trends and limitations lie ahead in the future? These tools need to adopt multiple working environments.

As Agile, Lean and other methodologies continue to be adopted beyond their original domains, an important question concerning the adaptability of supporting tools arises. Will these tools evolve to become more widely applicable, or will tools such as Microsoft Project primarily adjust to the varying requirements of projects

in different domains using Agile methodologies or ideology? Furthermore, it remains uncertain whether highly agile practices will be embraced more in contexts such as governmental work environments, given that the relatively rigid structure of Microsoft Project may not align well with the flexibility that such approaches demand. Or is the Agile implemented in those sectors significantly generalized? Rather than adopting Agile in its original form, organizations in sectors such as public administration may selectively adopt its principles, emphasizing only the features they desire. This suggests that what is often labeled as "Agile" outside the software industry may represent a hybrid or simplified version of the methodology, tailored to fit sector-specific constraints and institutional cultures. Such generalization raises questions about the extent to which its core values are preserved when translated into domains with fundamentally different or limited operational logic. This also further emphasizes and further proves that Agile may not only be a strict methodology, but also set of basic principles, a ideology or a culture. This also overcomplexifies the creation of tools. What is the direction that Jira or Microsoft Project takes in the future? Will these tools evolve toward greater specialization, catering to distinct domains and methodologies, or will they attempt to offer more generalized solutions capable of spanning multiple project types? At the moment, Jira provides multiple tools for software development. But could it fully support the needs of engineers, construction workers, finance sector and other domains through integrations similar to code collaboration tools, Git and CI/CD ? Furthermore, is universally best integration platform impossible in theory?

Interestingly, trends in integrations and functions of the tools can be observed across platforms. GitLab and GitHub combines code-related functionalities with a limited set of project management features, thereby serving primarily as a development-centric environment. By contrast, Jira and Microsoft's ecosystem of tools provide integrations that span nearly the full spectrum of project-related activities. This

raises an important question: why are code-related activities predominantly confined to platforms such as GitHub and GitLab, when broader project management is delegated to other tools? Could Jira, for example, be adapted to cater for a broader range of needs beyond its most dominant domains? There appears to be no significant research on the use of Jira in different domains. However, based on personal findings regarding job applications, Jira is used in many fields outside of IT in Finland. On the other hand, could GitLab and GitHub further extend their features regarding project management? Or is that unnecessary since tools like Azure DevOps and Jira exist?

SAP represents another example of an integrated tool ecosystem. While it is primarily an enterprise-level resource planning (ERP) platform, SAP also incorporates solutions for project and portfolio management alongside numerous other organizational functions. In contrast, tools such as Jira and GitLab offer more project related functionalities and are typically employed for more specific purposes such as task tracking, scheduling, code collaboration or project management. The SAP ecosystem, however, is designed to link project management with core business domains, including finance, human resources, and supply chain operations. Microsoft has also its own ERP / CRP tool known as Dynamics. What are the possibilities regarding integrations of all the organizational tools used in companies? How much automation and optimization is possible with the help of data-analysis and AI when all the functions can be seen under one software?

Security is another important aspect of AI-tool adoptation. As tools increasingly automate decision making and collect sensitive project data, ethical and regulatory issues will gain prominence. Data privacy, algorithmic transparency and bias mitigation in AI-driven settings will require some scrutiny. Organizations will need to implement governance frameworks to ensure responsible tool usage, especially in highly regulated industries such as healthcare, finance and government.

In conclusion the tools will need to adapt as the workflow, working environments, people and methodologies change. How will the AI tools change project management? Are there other revolutions ahead that could make significant changes to future needs of project management tools? For example Microsoft could further develop an integration platform with all the tools integrated into it. Managerial work could be done on Microsoft Project, but developer specific tasks could be seen in the platform even though they are done in Azure DevOps for example.

7 Conclusion

This thesis has explored the complex landscape of project management tools within software development using a dual-method approach involving practical application through a sample project the 'To-Do List Application', and empirical analysis via a survey. It provides a broad overview of how software teams engage with project management methodologies and tools in real-world contexts.

Rather than attempting to establish a definitive hierarchy among tools, this thesis emphasizes the contextual nature of tool effectiveness. It has shown that project management software cannot be ranked universally, but must instead be evaluated based on methodological alignment, organizational complexity, end product, and team-specific workflows. This insight is relevant to both software developers selecting tools and managers responsible for structuring development processes.

This thesis also highlights the growing importance of integration, configurability, and user adaptability. A consistent theme emerged from both the simulated sample project and the survey responses: no single tool can satisfy all functional and organizational requirements. This paves the way for hybrid tool adaptations which should allow more flexibility among teams.

The methodologies employed in this thesis have given some insights into the experimental and attitudinal dimensions of tool use, as well as providing qualitative and quantitative insights. This approach has enabled a nuanced understanding of how tools function and are perceived, adopted and adapted in practice. This thesis

offers both depth and breadth in its conclusions by simulating a software project with the lived experiences of over one hundred software professionals from different companies and backgrounds.

Naturally, some limitations must be acknowledged. While the simulated sample project offered a controlled environment for assessing tool functionality, it could not account for all the complexities of real-world development. These could include long-term team dynamics or evolving project scopes for example. Similarly, although the survey achieved a solid sample size, it was limited to the perspectives of the respondents who opted in. This may introduce bias. Furthermore, this thesis focused on only three tools, which limits the generalizability of the findings to the broader range of available platforms. The tools in question were also very different from each other. This may produce different results. The comparison was based on functionalities of tools rather than the branded tools itself. In addition to that, some alternative tools were highlighted and briefly compared and mentioned which gives this thesis slightly more breadth in its conclusions. Overall however, the chosen methodology offered a somewhat balanced and comprehensive approach to evaluating project management tools in software development. The combination of theory, practical experimentation and real-world feedback ensured some kind of understanding of tool performance and suitability in real-world contexts. However, the scope was limited due to the small number of tools and survey responses. While these constraints do not undermine the thesis's validity, they do limit its generalizability.

Ultimately, the aim of this thesis was to provide context for the selection of project management tools in software development. Project management tools should be evaluated not only on their features, but also on how well they fit into evolving team practices and organizational strategies. The ability to adapt to new tools remains essential when the environment is continually shifting regarding methodologies, technologies, project scope, expectations and standards. Effective

project management depends on more than just software capabilities or adherence to methodology. It is grounded in thoughtful and experience-based decision-making, and the ability to respond to complexity with agility and precision. This reaffirms the conclusion that effective project management is not solely a matter of methodology or software, but of informed, thoughtful, iterative, flexible, agile, improvable, adaptable, fast, reflective and extremely important part of work.

References

[1] F. W. Taylor, "Principles and methods of scientific management", *Journal of Accountancy*, vol. 12, no. 3, p. 3, 1911.

[2] F. W. Taylor, *Scientific management*. Routledge, 2004, ISBN: 9781134466238.

[3] J. Carruthers and A. Battersby, "Advances in critical path methods", *Journal of the Operational Research Society*, vol. 17, no. 4, pp. 359–380, 1966, Published by Taylor & Francis.

[4] D. L. Cook, *Program evaluation and review technique: Applications in education*. US Department of Health, Education, and Welfare, Office of Education, 1966, ISBN: 9780819106575.

[5] P. M. Institute, *A Guide to the Project Management Body of Knowledge: PMBOK Guide* (PMBOK® Guide Series). Project Management Institute, 2017, ISBN: 9781935589679.

[6] A. Khan, "Project scope management", *Cost engineering*, vol. 48, no. 6, pp. 12–16, 2006, American Association of Cost Engineers.

[7] P. Jovanovic and I. Beric, "Analysis of the available project management methodologies", *Management: Journal of Sustainable Business and Management Solutions in Emerging Economies*, vol. 23, no. 3, pp. 1–13, 2018.

[8] V. Rastogi et al., "Software development life cycle models-comparison, consequences", *International Journal of Computer Science and Information Technologies*, vol. 6, no. 1, pp. 168–172, 2015.

[9] H. F. Cervone, "Understanding agile project management methods using scrum", *OCLC Systems & Services: International digital library perspectives*, vol. 27, no. 1, pp. 18–22, 2011, Emerald Group Publishing Limited.

[10] D. J. Fernandez and J. D. Fernandez, "Agile project management—agilism versus traditional approaches", *Journal of computer information systems*, vol. 49, no. 2, pp. 10–17, 2008, Taylor & Francis.

[11] M. Ivanova, "Comparison between project management and software project management", *Machines. Technologies. Materials.*, vol. 8, no. 12, pp. 16–19, 2014, Scientific Technical Union of Mechanical Engineering" Industry 4.0".

[12] R. Hoda and L. K. Murugesan, "Multi-level agile project management challenges: A self-organizing team perspective", *Journal of Systems and Software*, vol. 117, pp. 245–257, 2016, Elsevier.

[13] N. B. Moe, T. Dingsøyr, and T. Dybå, "Understanding self-organizing teams in agile software development", in *19th australian conference on software engineering (aswec 2008)*, IEEE, 2008, pp. 76–85.

[14] T. Thesing, C. Feldmann, and M. Burchardt, "Agile versus waterfall project management: Decision model for selecting the appropriate approach to a project", *Procedia Computer Science*, vol. 181, pp. 746–756, 2021, CENTERIS 2020 - International Conference on ENTERprise Information Systems / ProjMAN 2020 - International Conference on Project MANagement / HCist 2020 - International Conference on Health and Social Care Information Systems and Technologies 2020, CENTERIS/ProjMAN/HCist 2020, ISSN: 1877-0509. DOI:

https://doi.org/10.1016/j.procs.2021.01.227. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050921002702.

[15] U. S. Senarath, "Waterfall methodology, prototyping and agile development", Jun. 2021. DOI: 10.13140/RG.2.2.17918.72001.

[16] B.-A. Andrei, A.-C. Casu-Pop, S.-C. Gheorghe, and C.-A. Boiangiu, "A study on using waterfall and agile methods in software project management", *Journal of Information Systems & Operations Management*, pp. 125–135, 2019, Romanian-American University, Scientific Research Department.

[17] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis", *arXiv preprint arXiv:1709.08439*, 2017.

[18] Instituteprojectmanagement org, *Agile methodologies and framework*, Accessed: 2025-05-28, 2025. [Online]. Available: https://instituteprojectmanagement.com/blog/agile-methodologies-and-framework/.

[19] K. Beck et al., *The agile manifesto*, Accessed: 2025-04-28, 2001. [Online]. Available: https://agilemanifesto.org/.

[20] P. A. G. Permana, "Scrum method implementation in a software development project management", *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 9, pp. 198–204, 2015, Science and Information (SAI) Organization Limited.

[21] Scrumorg, *Scrumteamagilemethodology*, Accessed: 2024-05-28, 2024. [Online]. Available: https://www.scrum.org/professional-scrum-competencies.

[22] D. K. Rigby, J. Sutherland, and H. Takeuchi, "Embracing agile", *Harvard business review*, vol. 94, no. 5, pp. 40–50, 2016.

[23] A. Kelly, *Changing software development: Learning to become agile*. John Wiley & Sons, 2008, ISBN: 9780470515044.

[24] J. Varga, Á. Csiszárik-Kocsir, and M. Garai-Fodor, "The development of modern-day competences in education, in the context of an agile approach", in *Frontiers in Education*, Frontiers Media SA, vol. 9, 2025, p. 1 485 273.

[25] M. Fowler, J. Highsmith, et al., "The agile manifesto", *Software development*, vol. 9, no. 8, pp. 28–35, 2001, [San Francisco, CA: Miller Freeman, Inc., 1993-.

[26] Y. Doz and M. Kosonen, "Governments for the future: Building the strategic and agile state", *Sitra Studies*, vol. 80, p. 18, 2014.

[27] K. Solala, *Kunnallishallinnon strategiatyön ketteryyden merkitys*, Bachelor's thesis, University of Tampere, 2025.

[28] A. Backlund, "Ketterät menetelmät ja niiden hyödyntäminen julkishallinnon organisaatiossa", M.S. thesis, University of Vaasa, 2024.

[29] L. Banica, M. Radulescu, D. Rosca, and A. Hagiu, "Is devops another project management methodology?", *Informatica Economica*, vol. 21, no. 3, 2017.

[30] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "Devops", *Ieee Software*, vol. 33, no. 3, pp. 94–100, 2016, Ieee.

[31] M. Loukides, *What is DevOps?* " O'Reilly Media, Inc.", 2012, ISBN: 9781449339104.

[32] F. Erich, C. Amrit, and M. Daneva, "Report: Devops literature review", *University of Twente, Tech. Rep*, 2014.

[33] R. Jabbari, N. Bin Ali, K. Petersen, and B. Tanveer, "What is devops? a systematic mapping study on definitions and practices", in *Proceedings of the scientific workshop proceedings of XP2016*, 2016, pp. 1–11.

[34] A. Putta, M. Paasivaara, and C. Lassenius, "Benefits and challenges of adopting the scaled agile framework (safe): Preliminary results from a multivocal literature review", in *Product-Focused Software Process Improvement: 19th International Conference, PROFES 2018, Wolfsburg, Germany, November 28–30, 2018, Proceedings 19*, Springer, 2018, pp. 334–351.

[35] Safe Framework Scaled Agile, *Safe scaled agile documentation*, Accessed: 2024-05-28, 2024. [Online]. Available: `https://framework.scaledagile.com/`.

[36] Safe framework, *Safe scaled agile documentation*, Accessed: 2024-05-28, 2024. [Online]. Available: `https://scaledagileframework.com/`.

[37] F. Almeida and E. Espinheira, "Large-scale agile frameworks: A comparative review", *Journal of Applied Sciences, Management and Engineering Technology*, vol. 2, no. 1, pp. 16–29, 1943.

[38] M. O. Ahmad, J. Markkula, and M. Oivo, "Kanban in software development: A systematic literature review", in *2013 39th Euromicro conference on software engineering and advanced applications*, IEEE, 2013, pp. 9–16.

[39] T. Björkholm and J. Björkholm, *Kanban in 30 days*. Packt Publishing Ltd, 2015, ISBN: 9781783000906.

[40] Y. Chhaya, *Ultimate Agile administration with Jira*. Orange education Pvt Ltd, 2023, ISBN: 978-8196782603.

[41] Gitlab, *Gitlab documentation*, Accessed: 2025-05-28, 2025. [Online]. Available: `https://docs.gitlab.com/`.

[42] S. Ailasmaa, "Release and deployment management in a scrum team : A case study", M.S. thesis, University of Turku, 2024.

[43] M. S. Arefeen and M. Schiller, "Continuous integration using gitlab", *Undergraduate Research in Natural and Clinical Science and Technology Journal*, vol. 3, pp. 1–6, 2019.

[44] C. Cowell, N. Lotz, and C. Timberlake, *Automating DevOps with GitLab CI/CD Pipelines: Build efficient CI/CD pipelines to verify, secure, and deploy your code using real-life examples*. Packt Publishing Ltd, 2023, ISBN: 1803233001.

[45] J. Fisher, D. Koning, and A. Ludwigsen, "Utilizing atlassian jira for large-scale software development management", Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), Tech. Rep., 2013.

[46] Atlassian, *Jira software documentation*, Accessed: 2025-04-28, 2024. [Online]. Available: `https://www.atlassian.com%20;%20https://confluence.atlassian.com/jira061`.

[47] P. Li, *Jira 8 Essentials: Effective issue management and project tracking with the latest Jira features*. Packt Publishing Ltd, 2019, ISBN: 978-1789802818.

[48] Microsoft, *Microsoft project documentation*, Accessed: 2025-05-28, 2024. [Online]. Available: `https://learn.microsoft.com/fi-fi/Project/`.

[49] M. Soni, *Hands-on Azure DevOps: CICD Implementation for Mobile, Hybrid, and Web Applications Using Azure DevOps and Microsoft Azure*. BPB Publications, 2020, ISBN: 9789389845341.

[50] J. Rossberg, "Agile project management with azure devops", *Apress, Berkeley, CA, USA, Tech. Rep*, 2019, Springer.

[51] R.-D. Ciure, "Enhancing project management reporting with power bi", M.S. thesis, Aalto University, 2024.

[52] R. I. Riti, A. C. Ionica, and M. Leba, "Transformative paradigms in it project management: A scholarly exploration of safe® and azure devops integration for enhanced innovation and strategic agility", *ENTRENOVA-ENTerprise REsearch InNOVAtion*, vol. 10, no. 1, pp. 347–359, 2024.

[53] J. K. Das, I. Elegbe, L. Coffie, R. Khadka, L. Chen, and Y. Ji, "Ai-powered it project management: Analyzing the effectiveness of advanced project management tools to ensure project efficiency", in *SoutheastCon 2025*, 2025, pp. 1554–1559. DOI: `10.1109/SoutheastCon56624.2025.10971718`.

[54] K. Shah and A. Trehan, "Streamlining software development: A comparative study of ai-driven automation tools in modern tech", *International Journal of Computer Engineering and Technology (IJCET)*, vol. 15, no. 6, pp. 1638–1650, 2024.

[55] J. C. Weng, "Putting intellectual robots to work: Implementing generative ai tools in project management", NYU SPS Applied Analytics Laboratory, Tech. Rep., 2023.

[56] M. H. Jarrahi, "Artificial intelligence and the future of work: Human-ai symbiosis in organizational decision making", *Business horizons*, vol. 61, no. 4, pp. 577–586, 2018, Elsevier.

[57] M.-C. Brad, F.-C. Birloi, A. Bratulescu, and I.-B. Blaga, "A comparative study of agile project management software tools", *Academy of Economic Studies. Economy Informatics*, vol. 16, no. 1, pp. 27–38, 2016, INFOREC Association.

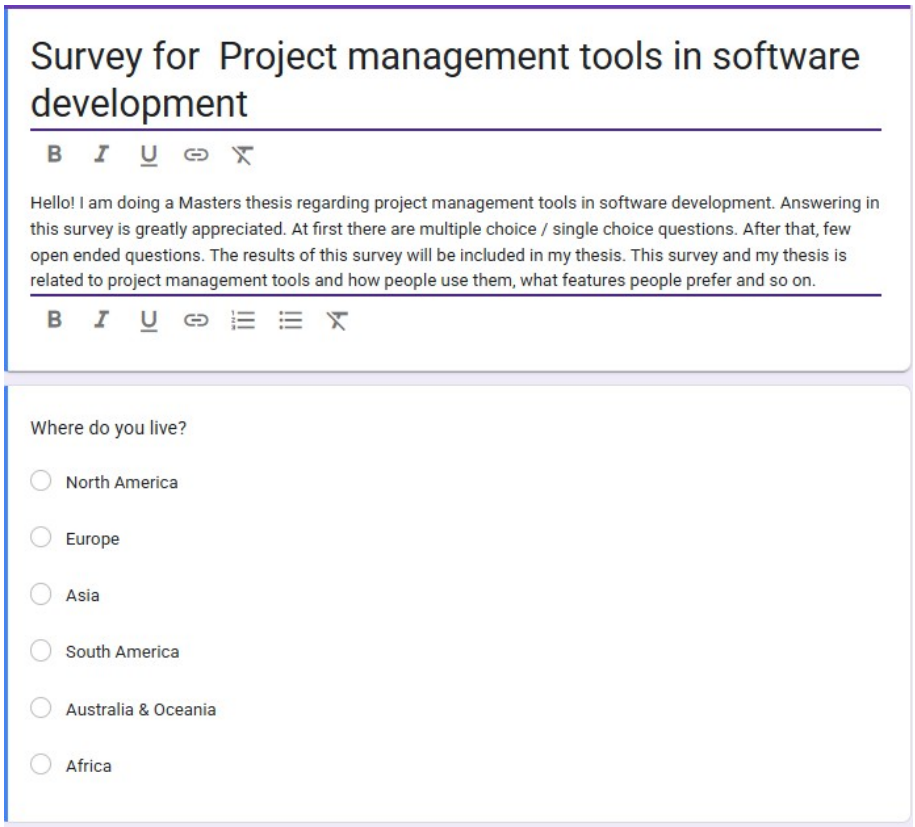
[58] G. Azizyan, M. K. Magarian, and M. Kajko-Matsson, "Survey of agile tool usage and needs", in *2011 agile conference*, IEEE, 2011, pp. 29–38.

[59] Versionone, *8th annual state of agile survey. versionone*, 2014. [Online]. Available: `https://www.learningtree.com/files/2013-state-of-agile-survey_lt.pdf`.

[60] S. H. Appelbaum, "Socio-technical systems theory: An intervention strategy for organizational development", *Management decision*, vol. 35, no. 6, pp. 452–463, 1997, MCB UP Ltd.

Appendix A Appendices

A.1 Survey screenshots



What industry your company is involved in? You can choose multiple options *

- [ ] Software development / IT services
- [ ] Financial services
- [ ] Cybersecurity
- [ ] Energy / utilities
- [ ] Government / Public sector
- [ ] Media / Entertainment / Gaming
- [ ] Manufacturing / Industrial technology
- [ ] Telecommunications
- [ ] Healthcare / HealthTech
- [ ] Muu: .....

What is your current role in a company *

- ( ) Developer
- ( ) Project manager
- ( ) QA engineer
- ( ) DevOPS engineer
- ( ) Scrum master
- ( ) Product owner
- ( ) Designer
- ( ) Upper management
- ( ) Muu: .....

How many years of experience do you have in software development? *

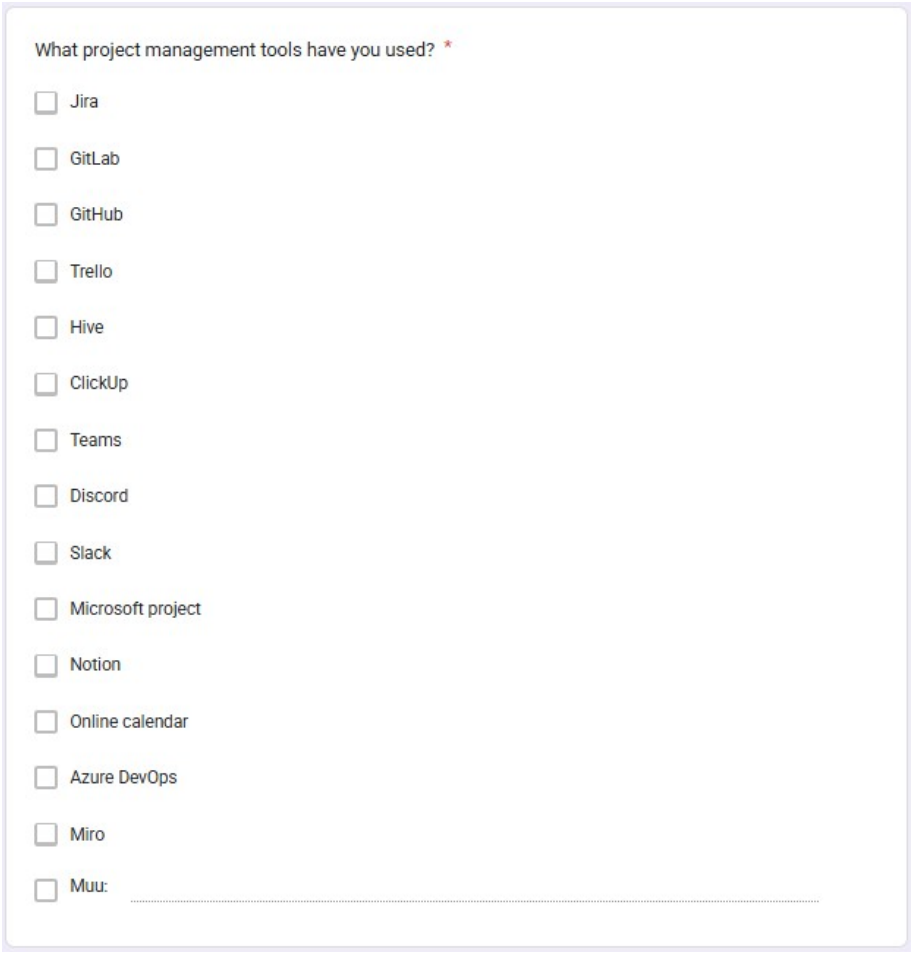
- ○ Less than 1 year
- ○ 1-3 years
- ○ 4-6 years
- ○ 7-10 years
- ○ More than 10 years
- ○ I have 0 experience (e.g. upper management)

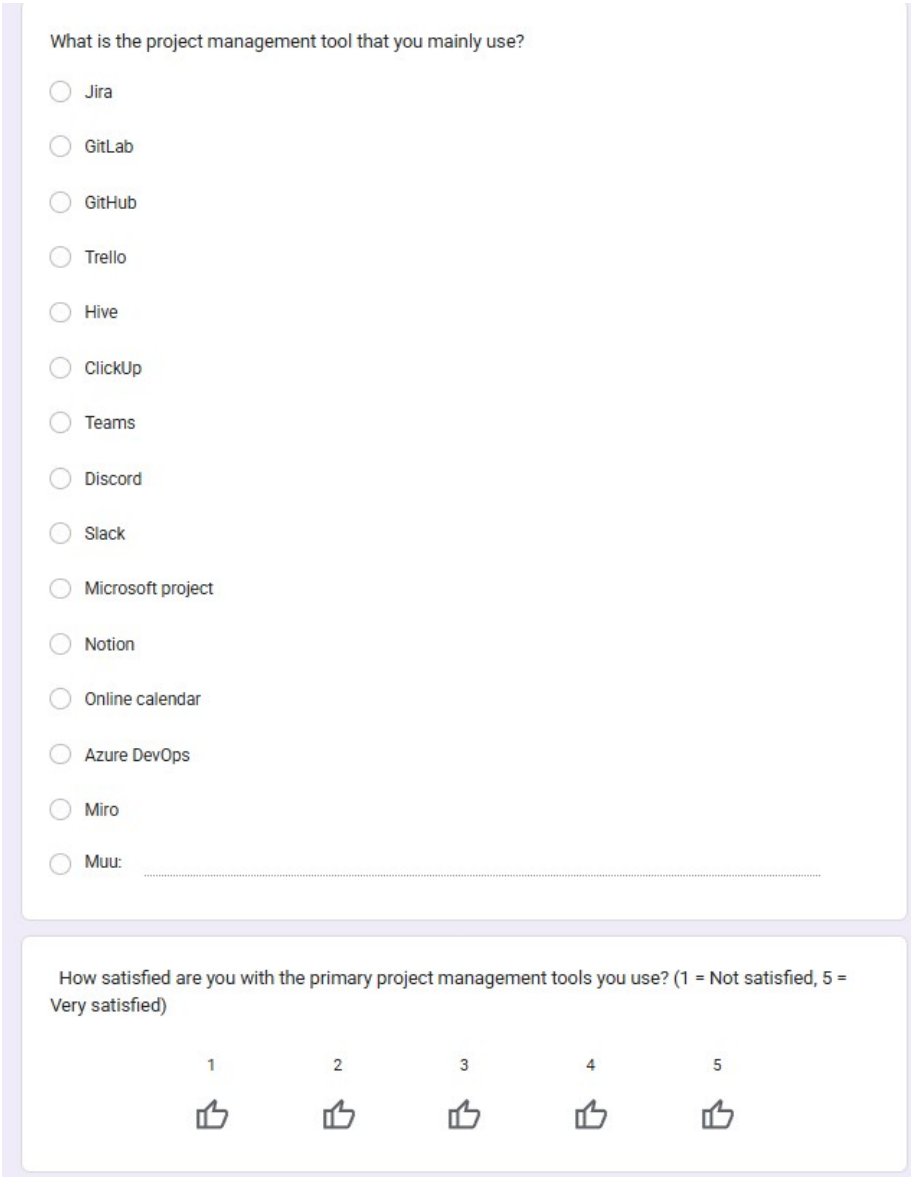
What is the size of your organization?

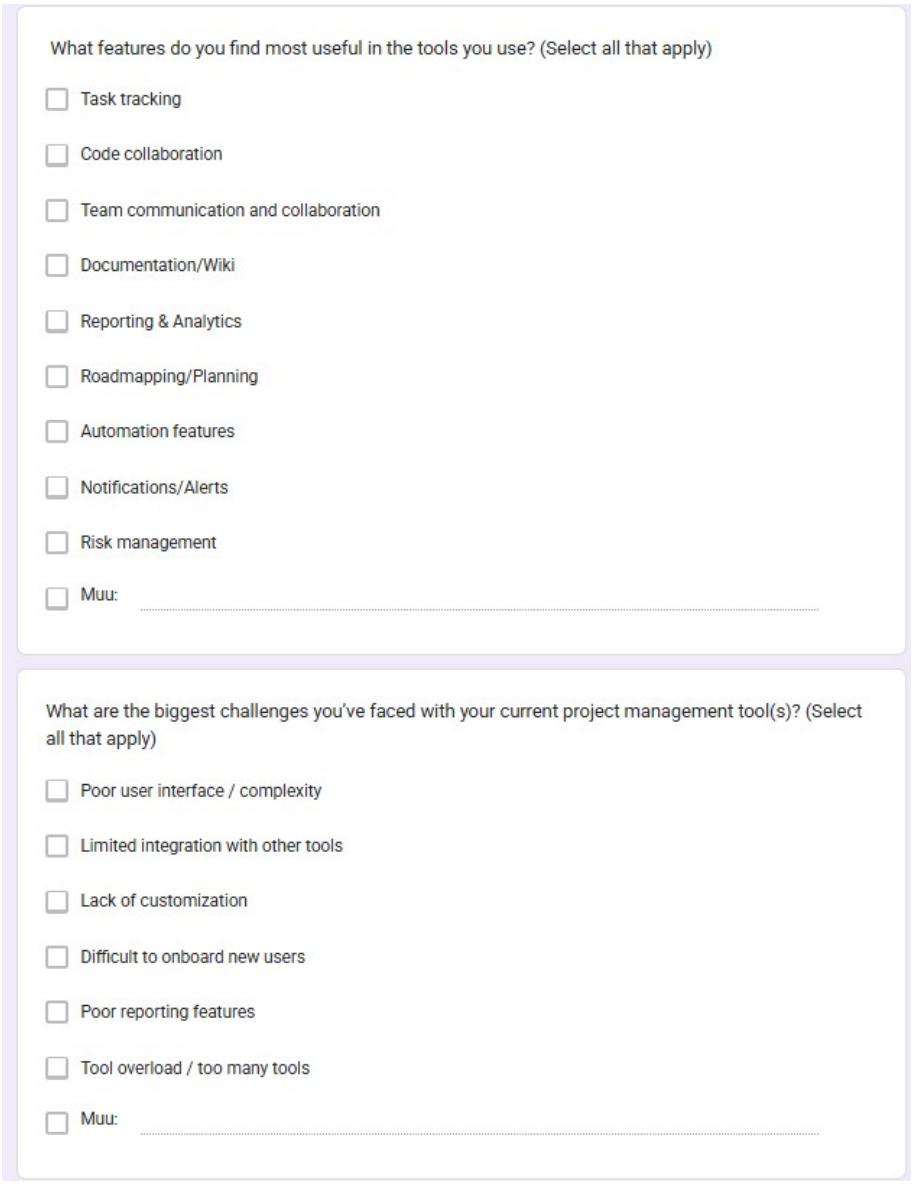
- ○ 1-10 employees
- ○ 11-100 employees
- ○ 101-500 employees
- ○ Over 500 employees

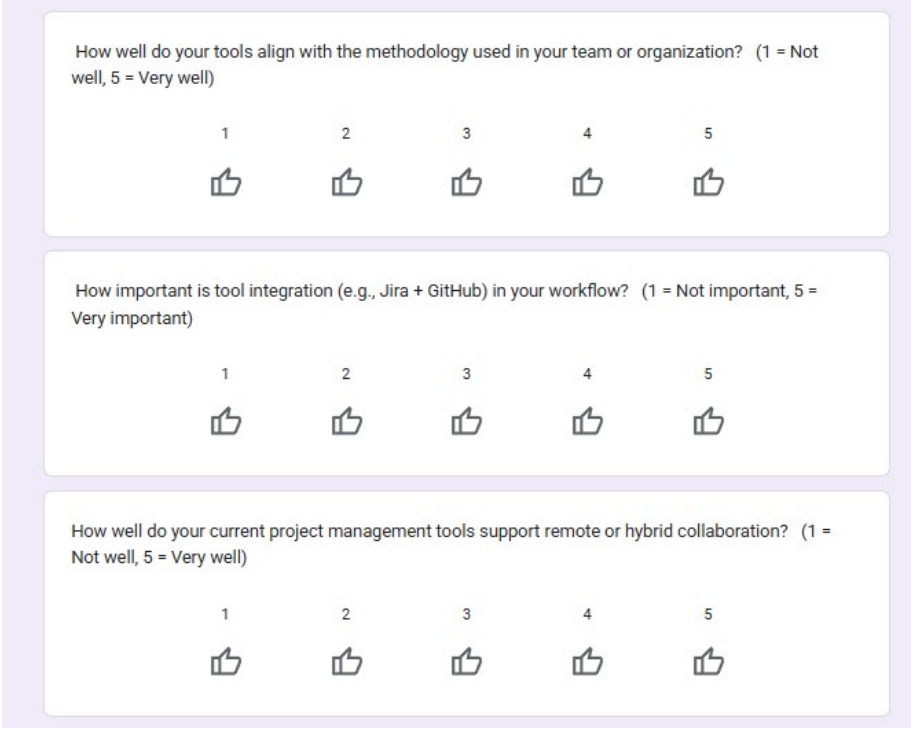
Which project management methodologies have you used in software development projects? (Select all that apply)

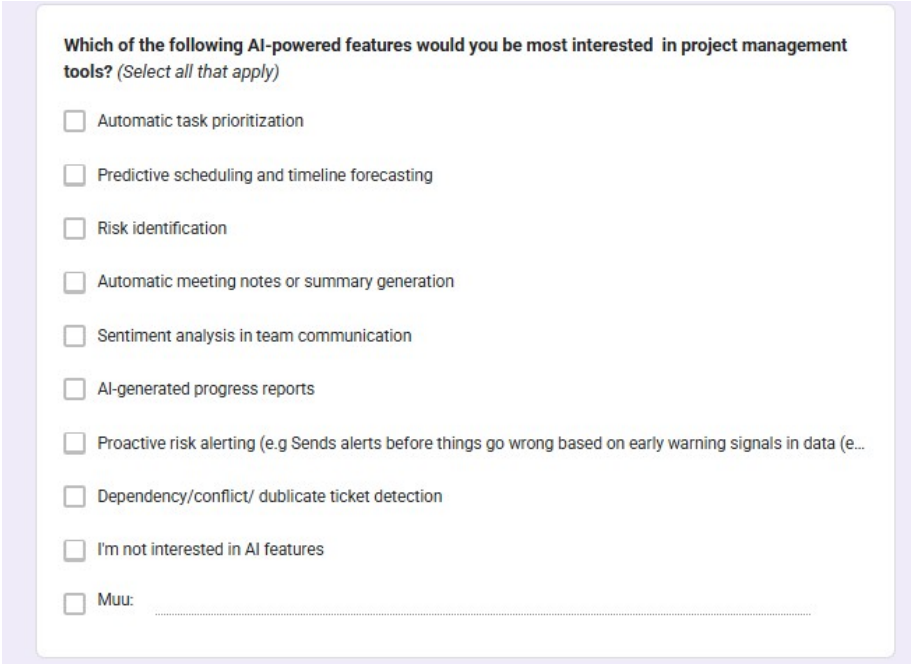
- ☐ Agile
- ☐ DevOPS
- ☐ Lean
- ☐ SAFe
- ☐ Prince2
- ☐ Waterfall
- ☐ Scrum
- ☐ Kanban
- ☐ I dont know what we use
- ☐ Muu: .....

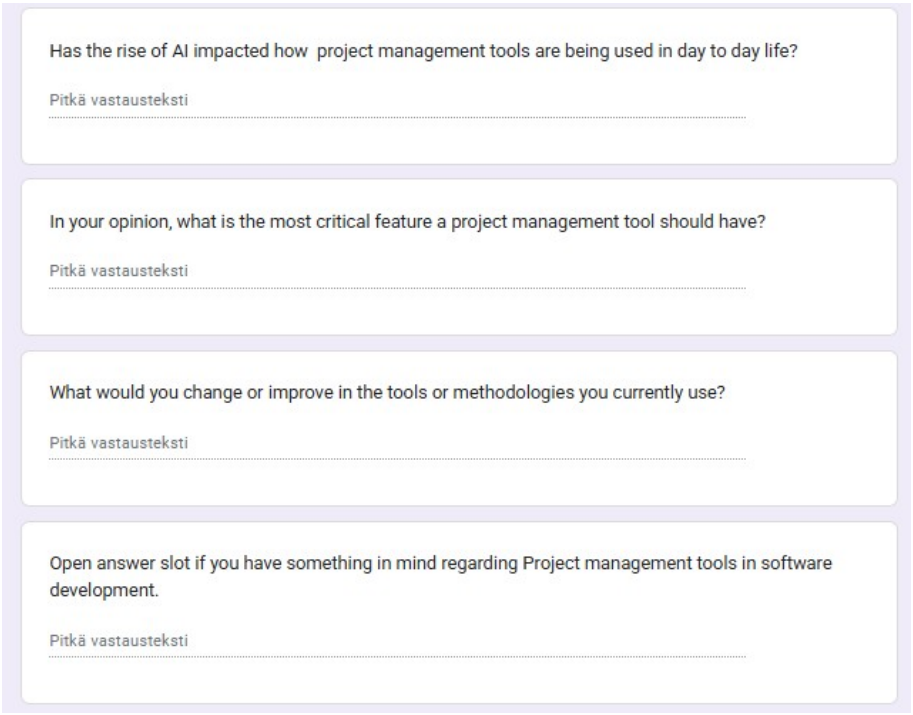

What project management tools have you used? *
Jira
GitLab
GitHub
Trello
Hive
ClickUp
Teams
Discord
Slack
Microsoft project
Notion
Online calendar
Azure DevOps
Miro
Muu:


What is the project management tool that you mainly use?
Jira
GitLab
GitHub
Trello
Hive
ClickUp
Teams
Discord
Slack
Microsoft project
Notion
Online calendar
Azure DevOps
Miro
Muu:
How satisfied are you with the primary project management tools you use? (1 = Not satisfied, 5 = Very satisfied)
1
2
3
4
5


What features do you find most useful in the tools you use? (Select all that apply)
Task tracking
Code collaboration
Team communication and collaboration
Documentation/Wiki
Reporting & Analytics
Roadmapping/Planning
Automation features
Notifications/Alerts
Risk management
Muu:
What are the biggest challenges you've faced with your current project management tool(s)? (Select all that apply)
Poor user interface / complexity
Limited integration with other tools
Lack of customization
Difficult to onboard new users
Poor reporting features
Tool overload / too many tools
Muu:


How well do your tools align with the methodology used in your team or organization? (1 = Not well, 5 = Very well)
1
2
3
4
5
How important is tool integration (e.g., Jira + GitHub) in your workflow? (1 = Not important, 5 = Very important)
1
2
3
4
5
How well do your current project management tools support remote or hybrid collaboration? (1 = Not well, 5 = Very well)
1
2
3
4
5


Which of the following AI-powered features would you be most interested in project management tools? (Select all that apply)
Automatic task prioritization
Predictive scheduling and timeline forecasting
Risk identification
Automatic meeting notes or summary generation
Sentiment analysis in team communication
AI-generated progress reports
Proactive risk alerting (e.g Sends alerts before things go wrong based on early warning signals in data (e...
Dependency/conflict/ dublicate ticket detection
I'm not interested in AI features
Muu:


Has the rise of AI impacted how project management tools are being used in day to day life?
Pitkä vastausteksti
In your opinion, what is the most critical feature a project management tool should have?
Pitkä vastausteksti
What would you change or improve in the tools or methodologies you currently use?
Pitkä vastausteksti
Open answer slot if you have something in mind regarding Project management tools in software development.
Pitkä vastausteksti