UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Karl Soosalu

Improving the sprint review and delivery process by enhancing the monitoring system

Master's Thesis (30 ECTS)

Supervisor:

Dietmar Alfred Paul Kurt Pfahl, PhD

Tartu 2025

Improving the sprint review and delivery process by enhancing the monitoring system

Abstract:

This thesis focuses on the development and implementation of a tool aimed at automating sprint summaries for the Data Services and Reporting (DSR) department at Playtech Estonia. The primary goal of the tool is to provide real-time insights into sprint performance by analyzing story point estimation accuracy, time logged versus estimated hours, and task completion rates. By integrating with Jira, the tool automates data collection and analysis, eliminating the need for manual data extractions and enabling continuous feedback loops. The development process involved using Java, Spring Boot, and Vue.js, and testing was conducted to ensure functional correctness and usability in real-world conditions. The tool features a dynamic and user-friendly dashboard that supports better visualization of key metrics. The results demonstrated the tool's ability to track dynamic changes in sprint data and offer valuable insights contributing to improved sprint planning and retrospective analysis.

Keywords:

Agile, Automation, Sprint Retrospective, Story Point Estimation

CERCS: P175 Informatics, systems theory

Sprindi ülevaate ja tulemuslikkuse parandamine monitooringusüsteemi täiustamise kaudu

Lühikokkuvõte:

Käesolev lõputöö keskendub tööriista arendamisele ja rakendamisele, mis on suunatud sprintide kokkuvõtete automatiseerimisele Playtech Eesti andmeteenuste ja raporteerimise (DSR) osakonnas. Tööriista peamine eesmärk on pakkuda sprintide kohta reaalajas ülevaateid, analüüsides töömahu hindamise täpsust, töötatud tundide ja hinnanguliste tundide erinevusi ning ülesannete täitmise määra. Jira integreerimise kaudu automatiseerib tööriist andmete kogumise ja analüüsi, eemaldades vajaduse käsitsi andmete eksportimise ja töötlemise järele ning võimaldades pidevaid tagasiside tsükleid. Arendamise protsessis kasutati Java, Spring Booti ja Vue.js, samuti viidi läbi testimine, et tagada funktsionaalne korrektsus ja kasutatavus reaalses töökeskkonnas. Tööriist sisaldab dünaamilist ja kasutajasõbralikku vaadet, mis võimaldab sprintide võtmenäitajate paremat visualiseerimist ja jälgimist. Tulemused näitasid tööriista suutlikkust jälgida dünaamilisi muudatusi sprintide andmetes ning pakkuda väärtuslikke ülevaateid, mis aitavad parendada sprintide planeerimist ja retrospektiivset analüüsi.

Võtmesõnad:

Välearendus, automatiseerimine, sprintide retrospektiiv, töömahu hindamine

CERCS: P175 Informaatika, süsteemiteooria

Table of Contents

1. Introduction

In modern software development, Agile methodologies have become the cornerstone of iterative, adaptable processes, ensuring teams can respond quickly to changes and continuously deliver value to stakeholders. Agile emphasizes collaboration, customer feedback, and small, frequent releases over rigid planning and documentation. Key frameworks under Agile, such as Scrum, are widely used for managing complex projects by breaking them down into manageable increments known as sprints. Each sprint includes planning, execution, and retrospective phases, where teams assess their work and look for ways to improve in the next iteration.

However, even with Agile's flexibility, teams can face challenges related to planning accuracy and tracking the quality of their estimates, particularly around story point estimation. Story points are a unit of measure in Agile used to estimate the effort required to complete a task. Getting these estimates wrong can lead to missed deadlines, over-commitment, and poor sprint planning.

In the case of the Data Services and Reporting (DSR) department at Playtech Estonia, a critical issue has emerged around the accuracy of story point estimations. The team lacks a clear and systematic method for evaluating whether the estimated story points align with the actual work completed. At present, there is no direct feedback loop for team members to assess their estimations against actual logged work, making it difficult to plan future sprints with confidence.

Currently, the team relies on exporting Jira data into Excel before and after sprints. While it provides a minimal level of insight, the method is manual, time-consuming, and prone to human error. Moreover, the analysis is often completed late, meaning that the sprint content is no longer fresh in the team’s memory. This delay hinders the ability to reflect accurately on the sprint's outcomes, analyze discrepancies between estimated story points and actual logged time, and implement timely improvements for subsequent sprints. Additionally, the approach lacks the capacity for systematic visualization or structured analysis, leaving much room for inefficiency and missed opportunities for process optimization.

The primary motivation for this thesis is to automate this process and provide real-time, actionable insights into estimation quality, helping the team improve their sprint planning and delivery processes. By developing a tool that offers a clear overview of logged hours vs estimated hours and completed story points vs estimated points, this research aims to provide the DSR department with a data-driven approach to sprint management. Ultimately, this improvement could help the team make better decisions about how much work to allocate in each sprint, minimizing the risk of over-committing or under-utilizing resources.

The primary aim of this thesis is to develop a tool that automates the generation of sprint summaries while emphasizing feedback on the accuracy of story point estimates. The tool will track and compare the initial story point estimates with the actual completed points to assess estimation accuracy. Additionally, it will analyze the time logged in Jira against the original estimates, identifying any significant discrepancies. This analysis will be presented through a visual dashboard that displays key metrics, such as story point completion rates, estimation accuracy across multiple sprints, and the comparison between logged and estimated hours. By implementing this tool, the team will have access to automated sprint review summaries that offer detailed insights into how well they estimate and complete tasks, facilitating continuous improvement in their planning and execution processes. Based on this historical data, the thesis will provide insights to assist teams in making more informed decisions for future sprint planning.

Achieving these goals is critical for several reasons. First, the tool will help the DSR department to better use Agile practices, aligning sprint outcomes more closely with initial planning efforts. This should result in better resource allocation and more realistic sprint goals. By improving the feedback loop on estimations, the team can become more efficient in delivering high-quality software, which is a key goal for any Agile team.

Moreover, this work is important because it addresses a common pain point in many Agile teams: the gap between estimated and actual task completion. While Agile promotes flexibility, accurate planning remains essential for maintaining steady project momentum and avoiding technical debt or missed deadlines. The tool will provide the necessary insights to close this gap.

The success of this thesis will be assessed based on several key factors. First, the tool must automate data collection from Jira, eliminating the need for manual updates or dependence on external systems like Excel spreadsheets. The dashboard should effectively present data related to estimation accuracy, comparing story point estimates with actual completion rates. This will allow the team to easily identify patterns and make necessary adjustments to improve their estimation practices. Additionally, the tool should be user-friendly, integrate seamlessly with existing systems like Jira, and be simple enough for the team to adopt without requiring extensive training.

At the outset of this project, the DSR team uses a basic, manual process for sprint analysis, relying on Excel exports from Jira to assess sprint performance. There is currently no automated or systematic method for reviewing the quality of their story point estimations or for comparing estimated vs actual work. Sprint retrospectives are carried out without detailed data analysis, limiting the team's ability to learn from past mistakes or improve their planning processes. This starting point highlights the need for a more sophisticated solution to automate these tasks, provide actionable insights, and help the team improve its overall sprint performance.

2. Related Work

This section explores the key literature surrounding Agile methodologies, story points, and the use of automation tools to evaluate estimations. The aim is to build a foundation for understanding the existing work on sprint planning, retrospectives, and process optimisation within Agile teams, which will serve as the basis for developing and evaluating the proposed tool in this thesis.

Agile methodologies, emphasising flexibility, adaptability, and iterative development, have become the cornerstone of modern software development practices. The Agile Manifesto introduced core principles that focus on collaboration, delivering working software frequently, and responding to change [1]. Since that, a significant amount of literature has focused on Agile in general, delving into frameworks such as Scrum, Kanban, and Extreme Programming. While these methodologies provide a structured approach to software development, one of their key challenges remains accurate effort estimation, particularly in Scrum-based sprint planning. Many studies explore core Agile concepts and challenges, covering sprint planning and retrospectives, which are widely adopted for managing tasks and tracking progress in software teams [2, 3].

Mallidi and Sharma [4] have examined various Agile story point estimation techniques used in Scrum, including Planning Poker, T-Shirt Size, Dot Voting, Bucket System, Large/Uncertain/Small, and Ordering Method. The concept of story points, a relative measure of effort for completing a task, is frequently discussed in Agile literature. Studies by Cohn [5] and others have emphasised how story points enable teams to estimate work without relying on absolute time-based measures, fostering better predictability and alignment. Research has also explored how teams calibrate story points during planning sessions and refine their velocity over time. For instance, Jorgensen and Molokken [6] provide insights into the factors influencing estimation accuracy, such as team experience, task complexity, and historical data. Zahraoui and Idrissi [7] introduced three additional factors—priority, size, and complexity—to enhance estimation accuracy. Meanwhile, Chongpakdee and Vatanawood [8] proposed a tool that estimates story points using document fingerprinting techniques to retrieve similar issues from JIRA. Despite these advancements, the core estimation process remains similar across methods and in most Agile teams. However, much of this work remains theoretical, focusing on how story points should be used rather than on their evaluation in practice. Despite their widespread adoption, there are limited studies examining whether story points are accurate indicators of effort, how consistently teams apply them, or how they correlate with actual time logged on tasks.

Few studies investigate the accuracy of Agile estimations at the task or sprint level. For example, Tanveer has also pointed out that the estimator's knowledge and experience and the change's complexity and impact are the main factors in the accuracy of the estimations [9]. Other research highlights the impact of over- or under-estimation on team productivity, project timelines, and stakeholder satisfaction [10]. While such studies acknowledge the importance of improving estimation accuracy, they do not systematically address tools or methodologies to evaluate or improve the process.

However, the effectiveness of Agile processes depends on how they are managed, making efficient sprint planning and retrospectives essential to the success of software development projects [11]. Despite Agile's widespread adoption, many organisations struggle to fully optimise these processes. Common challenges include improper backlog management, lack of alignment between teams, and poorly executed retrospectives. This indicates a clear opportunity for improvement by introducing process optimisations and automation.

There is a growing ecosystem of tools and plugins designed to support Agile teams, particularly those using platforms like Jira, Trello, Asana and Azure DevOps [12]. These tools often provide basic features for sprint planning, task tracking, and reporting. Jira, for example, includes plugins such as Advanced Roadmaps and Agile Reports to visualize team progress and velocity. Other third-party tools, such as EazyBI and Tempo Timesheets, allow more profound analysis of team performance and resource allocation.

Despite these advancements, few tools focus specifically on evaluating the accuracy of story point estimations. Most plugins offer retrospective views of velocity and burndown charts but lack granular analysis of discrepancies between estimated and actual effort [13]. While some tools allow teams to log work and compare it to estimates, the functionality is often limited to reporting rather than providing actionable insights. For example, EazyBI can generate reports showing story point completion rates, but it does not offer automated feedback mechanisms or recommendations for improving estimation practices. None of the tools provide deep customisation options to cover the needs of a specific team. As agile emphasises continuous improvement, a team needs to have the option to highlight discrepancies and offer insights into how estimation practices can be improved.

This section reviewed existing Agile estimation methods, studies on estimation accuracy, and available tools for sprint tracking. While existing research highlights the importance of estimation accuracy, few studies propose automated solutions for continuous feedback tailored to a team's unique requirements. This thesis aims to address the gap by developing a tool that evaluates the accuracy of story point estimations in Agile sprints according to the needs of a specific team. Unlike existing tools that primarily focus on tracking progress or generating reports, this solution will provide detailed feedback on estimation accuracy, enabling teams to refine their planning practices. By automatically collecting data from Jira and analysing discrepancies between estimated and actual work, the tool offers actionable insights for improving future sprints.

3. Background

3.1 Overview of Sprint Planning, Estimations, and Retrospectives

Agile methodologies, particularly Scrum, emphasise iterative development and continuous feedback. Within this framework, sprint planning is a crucial process that determines the scope and goals for each sprint, typically lasting two to four weeks. During sprint planning, the team collectively decides which product backlog items will be completed within the upcoming sprint. It assigns story points to these tasks, representing the effort or complexity required for completion. Estimating the workload accurately is critical to sprint success, as it enables the team to set realistic goals and ensure deliverables can be achieved within the allotted time. Story point estimation uses relative sizing to gauge the effort involved, but inaccuracies are common due to unforeseen complexities, unclear requirements, or changes in team velocity [4].

Sprint retrospectives are a core aspect of Scrum, and are also used in other Agile methodologies, held at the end of each sprint to reflect on what went well, what didn't, and how processes can be improved [14]. These retrospectives provide valuable insights for refining team processes, identifying recurring challenges, and improving the accuracy of future estimations. Despite their importance, retrospectives often suffer from a lack of actionable insights when key data, such as the accuracy of story point estimates, is not systematically reviewed [14]. Retrospective meetings often rely on qualitative feedback, leaving gaps in understanding the quantitative side of sprint performance.

One of the key challenges in sprint planning and estimation is the inherent uncertainty in software development. Factors such as incomplete requirements, technical debt, and unexpected dependencies can all contribute to deviations between estimated and actual effort [15]. Additionally, teams often face cognitive biases in estimation, including anchoring bias (being influenced by initial estimates) and optimism bias (underestimating effort) [16]. To mitigate these issues, some teams refine estimates based on historical velocity data or use relative estimation techniques, such as Planning Poker, to encourage discussion and consensus. However, without systematic tracking of estimation accuracy over multiple sprints, teams may struggle to identify recurring patterns of over- or under-estimation.

Existing solutions that help Agile teams manage sprints include Jira, Trello, and Asana, which offer basic tools for task management, sprint planning, and progress tracking [12].

While Jira, in particular, is widely used in Agile environments for its robust customisation options, it lacks native tools that automate the analysis of sprint estimation accuracy, making it challenging to provide concrete feedback on planning efficacy. Various Jira plugins, such as Jira Portfolio or Advanced Roadmaps, offer extended capabilities for managing multiple teams and long-term planning but do not directly address issues of estimation accuracy or logged versus planned work comparisons. In addition, manual approaches such as using Excel spreadsheets to track performance metrics can be used, but they are time-consuming and prone to human error.

Automating the collection and analysis of sprint data presents an opportunity to enhance estimation accuracy and retrospective effectiveness. By leveraging historical data, tracking discrepancies between estimated and actual effort, and providing teams with real-time feedback, automation can help Agile teams refine their planning and decision-making processes. Integrating such automated solutions within existing tools like Jira would enable teams to continuously improve their estimations, reduce inefficiencies, and ultimately enhance sprint outcomes.

3.2 Current Situation in the Playtech DSR Department

Playtech is one of the world's largest gaming software providers, reaching hundreds of millions of players globally by offering a wide range of platforms and end-to-end solutions. Founded in Estonia, the company's Estonian development centre remains one of its most important hubs. Playtech's primary business unit in Estonia is the IMS (Information Management System), through which operators gain complete oversight and control over player activities and system performance. The IMS platform differentiates Playtech from many competitors by providing advanced functionalities that enable operators to manage and monitor their platforms more efficiently [17].

The Data Services and Reporting (DSR) department, which this thesis focuses on, is a key part of the IMS, focusing on data reporting and flow management across critical systems. This department plays a vital role in supporting Playtech's operations by ensuring that data is correctly presented to regulators and stakeholders in a timely and accurate manner. The DSR department consists of 35 employees, structured into four development and one Quality Assurance (QA) team. The four development teams are the Event Platform (EVT) team, the Software Engineering (SE) team, the Reporting Engineering (RE) team and the Data

Management (DM) team. This collaborative structure enables the development of robust data solutions that meet the company's internal standards and regulatory requirements.

Maintaining high-quality standards is crucial in the context of regulatory demands and internal performance metrics. The DSR team operates within an Agile framework, ensuring iterative development and regular feedback through sprint planning, task management, and retrospectives. Sprint planning is a key part of this process, where team members estimate the complexity, effort, and time required for tasks. Story points, using Fibonacci numbers (1, 2, 3, 5, 8, 13, and 21), are employed to estimate these tasks in line with Agile practices, promoting consistency while avoiding over-analysis of task differences.

In this system, one story point roughly translates to four hours of work. For example, a task estimated at 2 points corresponds to a full workday, while a 13-point task represents a week's effort. The upper limit, 21 points, typically signifies a task requiring the entire sprint. These story points help the team gauge the scope of work for each sprint, ensuring that commitments align with the team's capacity and sprint duration.

While the DSR department uses story points to estimate task complexity, it's worth noting that the practice of mapping time directly to story points may not fully align with traditional Agile practices. In Agile methodology, story points are generally used to represent relative effort and complexity rather than translating directly into time or hours. However, the DSR team has chosen to adopt this approach because by correlating story points with time, the team can more easily manage workload expectations and plan sprints with a clear sense of how much effort is required. This also helps new employees better estimate their tasks as they can relate story point values to familiar time-based benchmarks. Also, this approach will provide the developed tool an option to measure the accuracy of their estimations, providing a more direct feedback loop on how well they're meeting their projected timelines.

However, the accuracy of these estimations depends heavily on past experiences and the availability of feedback on estimation precision. Without automated monitoring, the team lacks insights into how their initial estimations compare to the time spent on tasks. Despite using Jira for task tracking and backlog management, there is a significant gap in monitoring the quality of story point estimations. Currently, the department lacks a system to provide a clear overview of how well estimations align with actual work performed.

In the DSR department, there are also a few practices that can not be directly tracked by existing tools like Jira. For instance, certain Jira issues are flagged with a "LATE" tag during sprint planning. This tag is used to identify tasks that are not expected to be fully completed within the sprint timeframe but are anticipated to be initiated during the sprint. This approach helps in prioritising deliverables and managing workload expectations for tasks that are either large in scope or dependent on external factors, making their completion uncertain within the sprint. However, tracking such issues poses challenges as they can distort metrics like story point completion rates and time estimations, complicating the team's ability to assess sprint performance accurately. Integrating this nuance into the monitoring process is crucial for realistic sprint evaluations and continuous process improvement.

Another important factor not typically tracked by standard Agile tools is the involvement of multiple teams working on the same task during a sprint. For example, in the DSR department, both the SE and the QA team can contribute to the completion of tasks. Software engineers focus on the development aspect, while the QA team tests the functionality and ensures that the deliverables meet the required standards. Each team provides their input and effort, which usually requires separate estimations and time tracking. However, existing tools like Jira don't offer an integrated way to measure and compare the estimations and actual time spent by each team individually. This creates challenges in accurately assessing the full scope of work and performance for tasks that span multiple teams, making it difficult to evaluate how well each team's input aligns with the overall task completion and whether additional adjustments to estimations are needed.

Currently, the DSR team conducts retrospective meetings to review the sprint outcomes, but these discussions haven't often been based on clear evidence and qualitative feedback. There is no structured or automated system to track discrepancies between estimated and logged work or to analyse sprint performance in terms of story points. The team relies on manually exporting Jira data into Excel sheets to perform any analysis, which is cumbersome, prone to human error, and lacks scalability.

The absence of automated feedback on estimation accuracy has led to challenges in effective sprint planning. Without a clear view of past estimation errors, the team struggles to make informed decisions about the number of tasks or story points to plan for each sprint. This can result in either over-committing or under-committing, which can negatively impact team morale and project timelines. Furthermore, the reliance on manual Excel exports means that there is no consistent or real-time feedback loop for improving the sprint process.

Given this context, developing a tool to automatically generate sprint summaries and provide insights on story point estimation accuracy is essential. This tool would not only save the team significant time by eliminating manual data collection but also offer real-time insights into the accuracy of their sprint planning. By providing visual feedback on discrepancies between planned and actual work, the tool would help the DSR team make more informed decisions during sprint planning sessions, ultimately improving team efficiency and project outcomes.

4. Method

The primary goal of this research is to improve the sprint review and delivery process in the DSR department by developing a tool that enhances monitoring and feedback mechanisms. The tool will be an automated sprint monitoring and feedback system that integrates with Jira to track and analyze story point estimations, logged work, and sprint performance. This section outlines the approach and steps that will be taken to achieve these objectives. This methodology follows a structured, phased approach, as depicted in Figure 1, which outlines the main steps of the process alongside their respective inputs and outputs.

The methodology is divided into five main steps: System Requirements Definition, Design, Development, Deployment and Testing, Evaluation and Analysis, and Iteration Based on Feedback. Each step is interconnected and builds on the previous one, ensuring that the final tool meets both the functional and performance requirements. The design phase, for instance, will incorporate insights from the requirements gathering phase, while the development and deployment steps will ensure the tool is fully integrated with existing systems, such as Jira. Throughout this process, feedback loops will be implemented to refine and iterate the tool, ensuring that it meets the evolving needs of the DSR team.

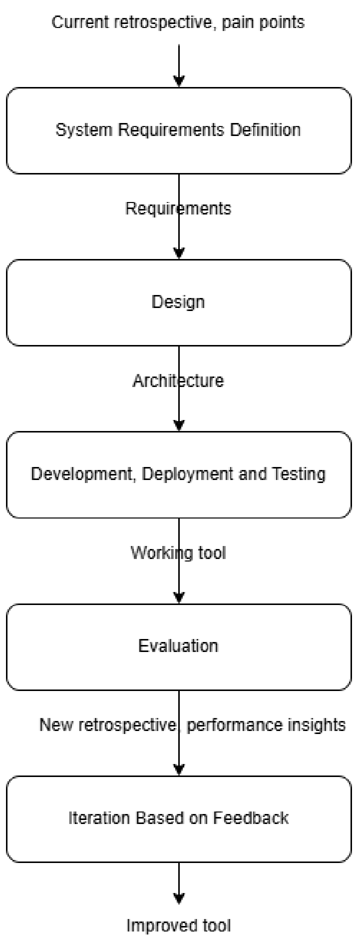


Figure 1. The methodology's structured approach details each step's inputs and outputs.

4.1 System Requirements Definition

The first phase focuses on identifying the functional and non-functional requirements necessary for the tool. This process builds on the challenges and inefficiencies highlighted in the background section. To elicit these requirements, I will conduct structured meetings and interviews with key stakeholders, including the project manager, delivery manager, and two product owners within the DSR team. These sessions will explore the current processes, tools, and pain points while gathering expectations for the tool's capabilities. Key metrics to be tracked—such as story point estimation accuracy, discrepancies between logged and estimated hours, and trends in sprint completion rates—will be identified collaboratively.

During these meetings, stakeholders will also provide input on their expectations for reporting features, the frequency and format of feedback, and any additional capabilities they believe could enhance their workflow. I will ensure that all feedback is captured and prioritized based on its importance and feasibility. This step will include creating a clear and actionable list of requirements, ensuring that each feature aligns with the objectives outlined in the thesis.

Additionally, the functional requirements, such as automated data collection, real-time feedback loops, and dashboard visualizations, will be defined. Non-functional requirements will also be outlined, including usability, scalability, and integration with tools like Jira. This comprehensive requirement-gathering process ensures the tool is tailored to the team's specific needs and provides a robust foundation for its development.

4.2 Design

The design phase involves translating the identified requirements into a blueprint for the tool's architecture and user interface. This includes creating visualizations of key metrics to make discrepancies in estimation accuracy and sprint performance easily interpretable by team members. The data flow and interactions between Jira and the tool are mapped to ensure seamless integration and efficient data retrieval. Furthermore, user-friendly interface designs are developed to encourage adoption and minimize the learning curve for team members. Focusing on both functionality and usability ensures that the tool delivers actionable insights while fitting naturally into the team's existing workflow. The outputs of this phase, such as architectural diagrams, serve as essential references for the subsequent development phase. The design process also involves selecting a suitable technology stack, including frameworks and libraries for both front-end and back-end development.

4.3 Development, Deployment and Testing

The third step focuses on building and deploying the tool in the DSR department's operational environment. The development process will use Java and Spring Boot for backend logic, with Maven as the build tool. The frontend will be developed using Vue.js, a framework well-suited for creating dynamic, user-friendly applications. PostgreSQL will be used as the database since the tool will be deployed in an environment where PostgreSQL is already present, ensuring compatibility with existing infrastructure and simplifying maintenance. All of these technologies are chosen to align with the company's existing technical stack, ensuring easier integration and maintenance within current processes. Integration with Jira's REST API will allow real-time retrieval of sprint data, ensuring automation of story point tracking and estimation accuracy analysis. Authentication with the Jira endpoints will be done via the Jira API access token.

The tool will be implemented as a standalone project and deployed as a separate Docker application, providing flexibility and modularity while leveraging similarities from existing solutions within the organization.

Multiple levels of testing will be applied to ensure high software quality. At the code level, unit tests will verify individual components' functionality and ensure their reliability. Initially, manual testing will involve comparing the tool's outputs to historical data from Excel sheets, ensuring consistency and accuracy. After deployment, early usage will help identify edge cases or unusual scenarios, allowing for iterative refinements.

The tool's automated functionality will include collecting and analyzing sprint data from Jira, enabling estimation accuracy tracking and time logging without manual intervention. These features are critical for generating sprint summaries and providing real-time feedback on performance metrics. Retrospective meetings will be conducted post-deployment to train team members on using the tool, ensuring a smooth transition. Feedback from these sessions will guide ongoing improvements to enhance utility and integration with the team's workflows.

4.4 Tool Evaluation

The evaluation phase focuses on assessing the tool's impact on the DSR team's sprint review and planning processes through qualitative insights. To gauge its usability and effectiveness, interviews will be conducted with key stakeholders, including project managers, delivery managers, and product owners. These discussions aim to capture feedback on how the tool has influenced decision-making and planning processes. Additionally, retrospective meetings will be observed to understand how the team incorporates the tool's outputs into their discussions and planning adjustments.

The feedback gathered from these methods will help identify both strengths and areas requiring improvement in the tool's design and functionality. Usability issues or recurring gaps in feature expectations will be documented and analyzed for potential iterations. Moreover, the qualitative feedback will serve as a foundation for understanding the tool's broader impact on team collaboration and alignment with sprint goals, ensuring its relevance and utility in the team's workflow.

4.5 Iteration Based on Feedback

The final step in the methodology involves iterating on the tool based on the feedback gathered from sprint retrospectives and evaluation. As the tool is used in real-world sprint planning and retrospectives, team members provide ongoing feedback on its effectiveness, usability, and any additional features they feel would improve its utility. Feedback will be collected during the sprint review and retrospective meetings, where I will be present to observe the tool's usage firsthand. Additionally, I will gather informal feedback from at least three team members, who will use the tool, to ensure diverse perspectives on its performance and to identify areas for improvement. This feedback, combined with data on how well the tool meets the team’s objectives, will drive iterative improvements to the tool’s functionality and interface. Changes may include refining data visualizations, enhancing integration with Jira, or adding new features to better support decision-making during sprint planning. Each iteration will be tested and deployed in a controlled environment, ensuring improvements align with team needs and enhance the tool’s value. This step establishes a continuous improvement cycle that ensures the tool evolves in response to changing team requirements, ultimately improving the sprint review and planning process.

5. Results

This section presents the thesis project's outcomes, detailing the sprint automation tool's development, implementation, and testing. It highlights how the system requirements were addressed, the tool's functionality, and its impact on the DSR department's workflow.

5.1 System Requirements Definition

The System Requirements Definition phase is essential for ensuring that the developed tool meets the needs of the DSR department at Playtech Estonia. This phase focuses on identifying the functional and non-functional requirements based on the challenges outlined in the background section. By collaborating with the DSR team, key metrics, features, and design considerations will be established to create a tool that directly addresses the inefficiencies of the current manual process. The requirements ensure the tool will effectively automate data collection from Jira, offer real-time feedback on estimation accuracy, provide a clear dashboard for analysis, and integrate seamlessly with existing systems. By addressing both technical and user needs, this phase lays a strong foundation for the design and development of a tool that will significantly improve sprint planning, execution, and retrospectives.

These requirements were derived through a series of structured meetings and interviews conducted with the project manager, delivery manager, and two product owners who are directly involved in sprint planning and retrospective activities. During these sessions, stakeholders were asked to share their experiences with the current process, pinpoint inefficiencies, and describe ideal improvements. Particular focus was placed on understanding the pain points in tracking story point estimations, generating summaries of time tracking data, and reviewing sprint outcomes. Their insights were instrumental in shaping the tool's feature set to ensure it addresses real-world needs and enhances day-to-day operations.

5.1.1 Functional Requirements

Functional requirements define the specific features and behaviours the tool must exhibit to solve the existing problems. These requirements are derived from the team's challenges with manual data collection, estimation accuracy, and sprint analysis.

5.1.1.1 Automated Data Collection from Jira

The tool must automatically gather specific sprint data for accurate analysis and reporting. This includes detailed information about each issue in the sprint, such as issue key, issue type, summary, status, story point estimates, and associated work logs. Additionally, the tool will track parent-child relationships for subtasks, aggregate logged work hours across subtasks, and distinguish between work completed during and outside the sprint period. By connecting to the Jira API, the tool will fetch this data for the active sprint and continuously update it to ensure real-time accuracy.

As the DSR team contains five smaller teams, then the tool should support tracking tasks worked on by multiple teams within the same sprint, such as software engineers and QA teams. It must distinguish between different contributors and their respective story point estimations and logged work to provide more accurate insights.

The tool should identify tasks not completed fully or partly within a sprint and track how their estimations evolve across multiple sprints. This will help analyze why certain tasks take longer than expected and whether adjustments in estimation practices are needed.

5.1.1.2 Estimation Accuracy Feedback

The tool must track and compare estimated story points with the completed ones on a sprint-level basis and track any discrepancies. The system should indicate if the task was completed within the sprint when tasks exceed the estimated time significantly, or when significant discrepancies between estimation and logged work occur. It should highlight trends in estimation accuracy across multiple sprints to help identify areas of improvement or recurring issues.

5.1.1.3 Time Logging vs. Estimated Effort

The tool should compare the total time logged for tasks across each team with the corresponding estimated effort and hours, identifying discrepancies where actual time spent is significantly higher or lower than estimated, providing a clear visualisation of this data. The tool must support this comparison on an individual task level and across the entire sprint.

5.1.1.4 Support for Historical Data Analysis

The tool should be able to track and display historical data for past sprints, allowing teams to identify long-term trends in estimation accuracy. The system must provide an option to compare past sprint data for retrospective analysis, showing improvements or regressions in estimation accuracy and task completion.

5.1.1.5 Integration with Existing Systems

Since the DSR department already uses Jira for sprint management, the tool must integrate seamlessly with Jira and the team's work practices without requiring significant changes to the existing workflow. The tool should be able to pull data directly from Jira through its API and update the dashboard and reports in real-time, reducing the need for manual input or third-party tools.

Additionally, it should be compatible with current team-specific conventions and logic, such as identifying and correctly handling issues tagged with LATE, ensuring that such custom practices are respected and reflected accurately in the reporting.

5.1.2 Non-Functional Requirements

Non-functional requirements describe the quality attributes the tool must possess to ensure it meets operational standards, including performance, usability, and scalability. These are critical for the long-term adoption and success of the tool.

5.1.2.1 Usability

The tool must be intuitive and easy to use, ensuring team members can quickly adopt it without extensive training. To evaluate this requirement, feedback on the interface's clarity, navigation, and overall design will be gathered through post-testing interviews. Key usability metrics, such as task completion time, error rate, and user satisfaction scores, will measure the tool's success in this area. The interface will prioritize a clean and straightforward design, adhering to established user experience principles to minimize cognitive load and improve the interpretability of data presented in the tool.

5.1.2.2 Scalability

The system should be able to track a larger volume of tasks or sprints as the department grows, ensuring the tool remains useful over time. Scalability will be tested by simulating increasing data loads, such as larger sprints with more issues, and evaluating the tool's performance under these conditions. Metrics like response time and data processing speed will be measured to ensure the system can handle the additional load effectively.

5.1.2.3 Performance

The tool must operate with minimal latency, providing real-time insights into sprint performance. Data retrieval from Jira should be quick and efficient, even for larger sprints with many issues and worklogs. Performance will be validated through stress testing and benchmarking, where the tool's response time is measured. Tests will ensure that even with large datasets, the system remains responsive.

5.2 Design

The Sprint Tracker system is an application designed to address challenges in sprint monitoring and retrospective analysis by integrating directly with Jira. It automates data collection, tracks story point estimation accuracy, and provides insights for improving sprint planning. Below is a detailed description of its design and components.

5.2.1 Database Structure

The application uses a relational database to store and manage data related to sprints, issues, and team members. The database is organized into three main tables: sprint, jira_issue, and team_member. The sprint table captures information about individual sprints, including fields such as sprint_id (primary key), name, state (active or closed), start_date, and end_date. The jira_issue table stores details about issues associated with each sprint, including a composite primary key (issue_id and sprint_id), title, status, type, team, story point estimations, logged hours, and flags for late or adjusted estimates for each team (EVT, SE, RE, DM, QA). Finally, the team_member table maintains a list of team members, identified by id (primary key), with additional fields jira_username and team. These tables are linked through foreign key relationships, with sprint_id serving as a common field between sprint and jira_issue. This structured approach ensures data integrity, efficient querying, and seamless integration with the application's business logic.

5.2.2 Business Logic

The business logic of the application is built around a scheduled, automated workflow that integrates with Jira to manage and update sprint data (see Appendix I). The core functionality is handled by the SprintUpdateService, which runs at hourly intervals. The service first retrieves the current active sprint, checking its status through the JiraService, which communicates with Jira’s REST API. If the current sprint is no longer active or there are no active current sprints in the database, then the service fetches the latest active sprints and updates the database accordingly.

Once the active sprint is confirmed, the system fetches all issues associated with the sprint. For each issue, the service retrieves work log data from the Jira API, filtering it by team members and sprint dates. The worklogs are aggregated for each team member and issued to calculate the total logged time per team. The SprintUpdateService updates the corresponding issue records in the database with these details, ensuring that all the data is accurate and up-to-date.

The system will have an error-handling mechanism to ensure that the application behaves gracefully if an error occurs during the data-fetching process from Jira. For example, if Jira's API is unavailable, will log the error and retry after one hour. This will make the tool more resilient.

5.2.3 Provided API endpoints

The application backend provides a set of RESTful APIs through its controllers to facilitate interaction between the user interface and the database (see Appendix II). The SprintController exposes endpoints such as GET /api/sprints to retrieve all sprints and GET /api/sprints/{sprintId}/tasks to fetch detailed information about tasks within a specific sprint. These endpoints ensure that the application dynamically provides up-to-date sprint and task data as requested by the UI.

Similarly, the TeamMemberController handles team member management, offering endpoints like POST /api/teammembers to add a new team member and GET /api/teammembers to retrieve a list of all team members from the database. Each request is routed through the controller to perform the necessary database operations and return the requested data in a structured format.

These APIs enable seamless communication between the UI and backend, ensuring that all operations, such as fetching sprints, tasks, and team members or adding new ones, are efficiently managed while maintaining a clean separation between the application layers.

5.2.4 User Interface

The user interface is designed to provide a clear and structured view of sprint progress, enabling teams to easily analyze their estimations and logged work. The UI follows a dashboard-based layout, prioritizing usability and clarity. It allows users to select a sprint, filter results by team, and view key sprint performance metrics intuitively.

At the top of the interface, a sprint selection dropdown enables users to choose from available sprints. This selection dynamically updates the displayed data to reflect the chosen sprint's details. Below this, a team selection panel with checkboxes allows users to filter results by specific teams (EVT, SE, RE, DM, QA). If no teams are selected, the dashboard defaults to displaying aggregated data for all teams, providing an overview of the entire sprint.

For each selected team, the interface presents summary metrics in a structured format. It displays the total story points planned at the start of the sprint compared to what was actually completed, along with a breakdown of tasks whose estimations were adjusted late in the sprint and their corresponding completion rate. Additionally, it shows the sum of all time logs recorded by team members during the sprint, allowing teams to evaluate workload distribution.

Below the summary metrics, a detailed task table lists all sprint issues, providing granular insights into individual tasks. Each row in the table includes the task title along with its Jira issue ID, which is displayed as a clickable link leading directly to the corresponding issue in Jira. The table also includes information about the assigned team, task status (To Do, In Progress, Done), estimated and completed story points, total logged hours, and indicators for late estimation or late completion.

The UI follows a clean, minimalistic design to ensure readability and reduce cognitive load. Data visualization elements, such as colour-coded indicators, help users quickly interpret key insights. Completed tasks are marked green, making it easier to spot areas requiring further discussion. By combining a structured filtering system with real-time sprint data, the UI ensures that users can efficiently track team performance and make informed decisions for future sprint planning.

5.3 Development, Deployment and Testing

The development, deployment, and testing phases of the tool have been completed. This process ensured that the system met all functional and non-functional requirements while providing a seamless user experience. The following sections outline the key aspects of each phase.

The backend was implemented using Java and Spring Boot, providing a robust and scalable foundation for handling Jira API interactions and data processing. The database layer was built using PostgreSQL, ensuring efficient data storage and retrieval for sprint-related metrics. The frontend was developed using Vue.js. Meaning that everything went according to the design.

The UI components were designed with usability in mind, incorporating clear data visualizations and structured layouts to enhance sprint tracking and analysis. The development process also involved integrating the system with Jira's REST API, allowing real-time synchronization of sprint data, including story points, time logs, and issue statuses.

In general, everything went according to the initial design. Each prominent feature was implemented, and automated unit tests were written to ensure the correctness and maintainability of core functionality.

The application was containerized using Docker, ensuring a consistent and portable deployment across various environments. Each new build was assigned an incremental version and pushed to Playtech's Artifactory repository. For deployment, the existing server hosting the PostgreSQL database was utilized. The latest application image was pulled from Artifactory and deployed on this server, ensuring seamless integration with the database and maintaining version control for efficient updates.

The testing phase focused on ensuring that the tool performed as expected in real-world conditions. Testing was carried out in multiple stages, including unit testing and user acceptance testing.

Unit testing was conducted for backend services, validating that individual components, such as Jira API interactions and internal calculations, functioned correctly. For example, extracting story points from custom Jira fields and ensuring they are correctly parsed for further analysis. User acceptance testing involved real users from the DSR department interacting with the tool in a controlled environment. Testing was performed iteratively over the course of several sprints, during which the tool was actively used in sprint retrospectives. Feedback was gathered during these sessions to identify usability issues, inconsistencies in data, or feature gaps. No major bugs related to the functional requirements were detected during testing. However, a few data inconsistencies and edge cases were identified and addressed during the tool iteration phase.

In addition to functional validation, the tool was also evaluated for usability, scalability, and performance. Usability testing focused on ensuring that users could intuitively navigate the interface and understand the displayed metrics. This was assessed through direct observation during retrospectives and informal feedback sessions, leading to minor layout adjustments. The tool's performance was evaluated based on response time and reliability. Results showed that the tool maintained optimal performance, retrieving and processing sprint data instantly, even for sprints with a high number of tasks and worklogs. Scalability was verified by testing the tool with large datasets in local environment, containing hundreds of issues per sprint.

Overall, the development, deployment, and testing processes ensured that the tool is a reliable and effective solution for sprint monitoring and retrospective analysis. The tool is now fully operational and integrated into the team's workflow, enhancing decision-making and sprint planning based on real-time and historical data. Screenshots of the deployed tool are visible in Figures 2 and 3.

The application source code, with Playtech-specific details removed, is available in a GitHub Repository[1]. Some configuration aspects of the application are hidden and the application cannot be run independently, as it relies on proprietary Playtech data retrieved directly from Jira, which is not accessible outside the organization. Furthermore, it is not possible to access the live running version of the application, as it is deployed internally within Playtech's infrastructure.

[1] https://github.com/karlsoosalu/sprint_tracker

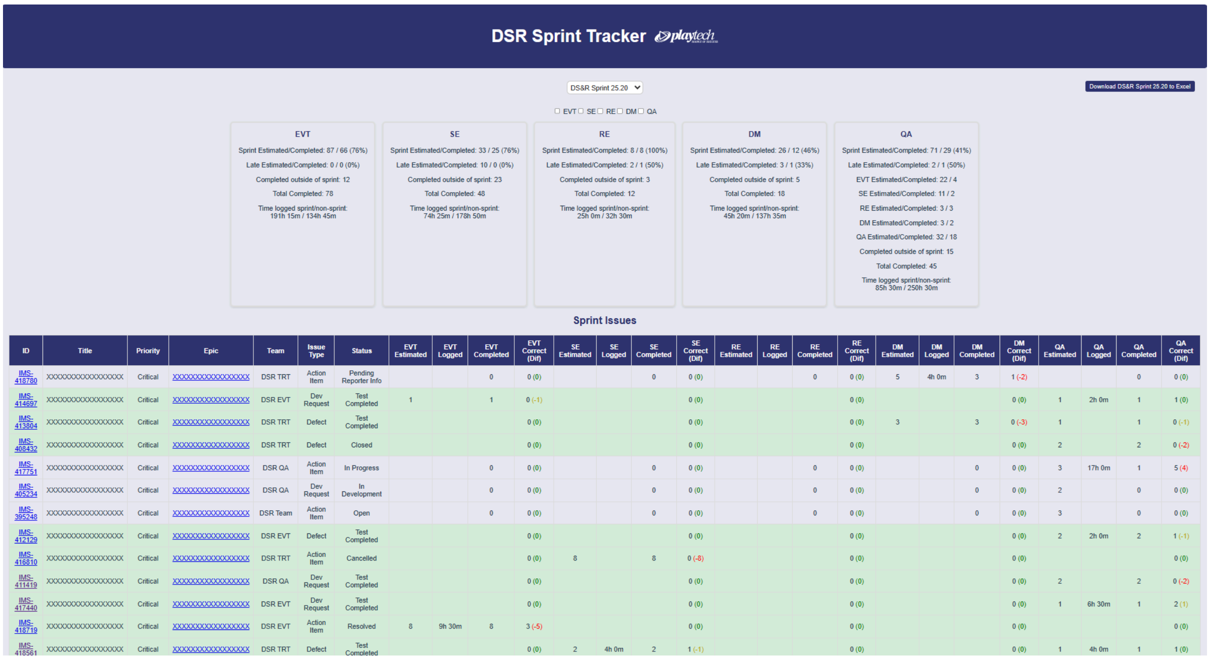


Figure 2. Screenshot of the deployed tool without filters applied.

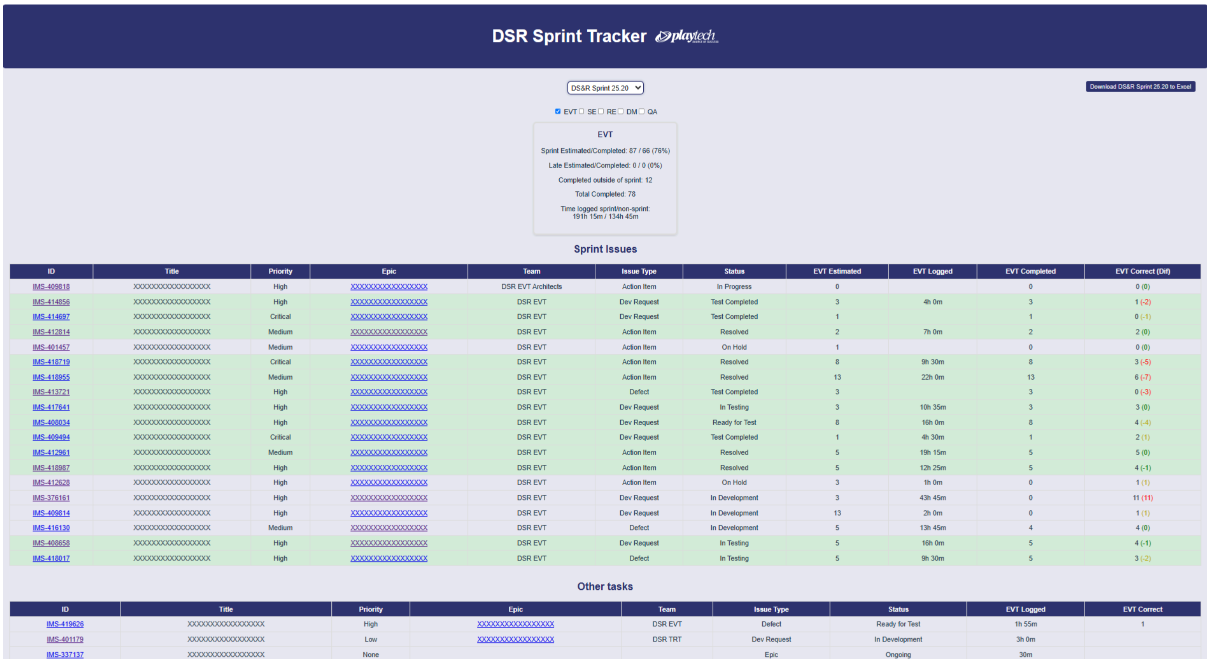


Figure 3. Screenshot of the deployed tool with applied filter.

5.4 Tool Evaluation

The evaluation phase of the Sprint Tracker tool was conducted to assess its effectiveness and impact on the DSR department's workflow. The evaluation process involved gathering feedback from the DSR team members, analyzing key performance metrics, and validating the tool’s functionality against the defined requirements.

Initially, there was a plan to compare manually constructed Excel data with the tool-generated data to validate its accuracy. However, this step was ultimately skipped because the data structures differed slightly, and the consensus was that the tool presented the information in a clearer and more useful way. Additionally, the team preferred the automated approach and did not want to spend extra time manually downloading and analyzing the same data in Excel anymore, as the tool effectively streamlined the sprint analysis process. After that the tool was just monitored and explored against Jira.

To assess the tool's ease of use, feedback was collected from team members through interviews, calls and messages. The primary focus was on its ability to support sprint planning and retrospective meetings and the tool's ease of use and clarity of displayed data. Users reported that the intuitive interface, featuring a sprint selection dropdown and optional team filters, made it easy to retrieve relevant sprint data. Additionally, the clear visualization of estimated versus completed story points and logged hours significantly improved the sprint review process.

Team leads highlighted the value of displaying the tasks that were not completed in the sprint and especially the feature to display, which tasks were over or under estimated, so the teams were able to make analysis based on that.

The introduction of the Sprint Tracker tool significantly improved the efficiency of sprint planning and retrospectives. Previously, manual data collection from Jira was time-consuming and prone to errors, making it difficult to assess estimation accuracy and team performance trends. With automated data retrieval and real-time updates, teams could focus more on analysis rather than data compilation.

5.5 Iteration Based on Feedback

During the development, testing, and evaluation phases, continuous feedback was collected from users to refine and optimize the tool's functionality. Feedback was primarily gathered from team members who directly interacted with the tool during sprint planning and retrospectives. Specifically, informal feedback was solicited from five active users, while I also participated in both the team's internal retrospective meetings and the broader retrospective meetings where team leads were present. These sessions provided valuable insights into how the tool was perceived and used in real sprint analysis discussions. Based on this input, several enhancements were made to better align the tool with the team’s specific needs and improve overall usability.

One of the significant changes was the adjustment of how sprint data was captured. Initially, the tool did not mirror the Excel-based sprint tracking, which recorded the sprint's state at its start. Based on feedback, the tool was changed to do the same. Initially, tool was designed to capture the most up-to-date sprint data, meaning new issues added during the sprint were automatically incorporated, whereas the Excel-based analysis only recorded the state of the sprint at its start. This was changed in a way that if new tasks are added during the sprint, then they would not appear in the sprint as planned tasks, but are instead categorized separately as other tasks, where team contributed. This way the tool could be used more to reflect the quality of the planning.

Another major enhancement involved filtering sprint data to exclude port tickets, which were irrelevant to the team's sprint evaluation. Port tickets refer to tasks that involve porting changes from the main branch to other release branches without requiring significant new development effort, as the main work is already done in the original ticket.

Additionally, the QA team overview was refined to offer a more granular breakdown of work completed within different development teams, improving visibility into testing contributions. This helped identify how QA efforts were distributed across various areas of the product, enabling better resource allocation, performance tracking, and earlier detection of bottlenecks.

Further improvements included the addition of two new columns: "Priority," which provides insights into the criticality and importance of tasks within a sprint, and "Epic," which shows the larger initiative or feature that each ticket is related to. Additionally, an Excel export feature was integrated, enabling users to download sprint data for offline analysis and reporting. These refinements made the tool more adaptable and user-friendly, ensuring it effectively supported sprint planning and retrospective analysis.

To improve sprint summary clarity, a percentage calculation was added to the estimated vs. completed story points summary. A difference indicator (diff) was also introduced in parentheses for the "correct" estimation column, and this value was visually color-coded, green for correct estimations, yellow for estimations off by one, and red for differences of two or more steps off in the Fibonacci-based estimation system that is used by the team.

Time tracking was also improved. First, support was added for tracking time logged after the sprint had ended, ensuring that logged hours for the previous sprint are still captured for two weeks. Second, the logic was adjusted to count time entries where the log timestamp is strictly less than the sprint end time, aligning with how Jira marks the sprint end time as the first day of the next sprint.

In terms of data processing, estimation parsing was enhanced by ignoring any additional details included within parentheses, allowing users to append context to their estimations without affecting the calculation logic.

The implementation of the team_members database table and its related API endpoints was mostly completed, with backend development finalized but the UI components still pending. However, after deploying the tool to the server, it was discovered that the team member information was already stored in the existing PostgreSQL database for other use cases, making it more efficient to reuse the existing data rather than maintaining a separate implementation. This approach simplified the system, reduced redundancy, and ensured consistency across different applications utilizing the same team member information.

6. Discussion

The development of the tool highlighted the importance of automating as much work in today's business as possible, in the context of this thesis the sprint tracking and estimation analysis to improve team efficiency and planning accuracy. By integrating directly with Jira, the tool eliminated the need for manual data extraction and analysis, significantly reducing the time spent on sprint review preparations. The iterative development approach, incorporating feedback from the team, ensured that the tool addressed real-world challenges faced in sprint monitoring. Additionally, the transition from static Excel-based analysis to a dynamic, real-time tracking system provided the team with more actionable insights, improving their ability to plan and evaluate sprints effectively.

One key takeaway from the project was the shift from manually maintained Excel sheets to a dynamic, real-time data analysis approach. Initially, there were concerns about how the tool's data compared to manually recorded Excel data, leading to a planned validation phase where outputs from both sources would be compared. However, after deployment, the consensus was that the tool presented the data in a clearer and more useful way, eliminating the need for this validation step.

Another valuable insight was the necessity of adapting the tool to align with existing workflows. Early iterations of the tool included data points that were not as relevant to the team's sprint review process, such as port tickets. Based on feedback, these were excluded to prevent unnecessary noise in the reports. Additionally, enhancements were made to the QA team overview to provide a more granular breakdown of their contributions within each development team. These adjustments demonstrated that even well-planned initial designs must remain flexible and responsive to user needs.

Similarly, the decision to drop the implementation of team member data storage and related API endpoints was an example of resource optimization. Initially, the tool included a separate system for managing team member information; however, after deployment, it was discovered that the same data was already being stored in the PostgreSQL database for other use cases. Rather than duplicating functionality, it was decided to reuse this existing data, reducing redundancy and simplifying the system architecture. This decision reinforced the importance of evaluating how new tools fit within an organization's existing infrastructure.

Overall, the development and deployment of the tool provided valuable lessons in agile software development, team-driven iteration, and system optimization. The feedback-driven enhancements ensured that the tool not only met its initial requirements but also adapted to the evolving needs of the team, demonstrating the effectiveness of continuous improvement in software projects. By prioritizing automation, integration with existing systems, and iterative refinement based on real-world usage, the tool successfully addressed the core challenges faced by the team, ultimately contributing to a more efficient sprint review and planning process.

7. Conclusions

This thesis successfully addressed the challenge of improving sprint review and estimation accuracy within Playtech Estonia's Data Services and Reporting (DSR) department. By developing an automated tool for analyzing sprint data, the project replaced the previous manual and error-prone Excel-based approach with a more efficient and reliable system. The tool provided real-time insights into story point estimations, actual work completed, and logged time, allowing the team to assess their estimation accuracy and improve future sprint planning. The implementation of a dynamic, user-friendly dashboard enhanced the visualization and analysis of sprint performance, ensuring that critical data remained easily accessible and consistently up to date.

The requirements were gathered, the design was crafted, the tool was developed, deployed, and tested, and it was subsequently evaluated and refined based on user feedback. The primary objective of this thesis was to create a solution that integrates seamlessly with Jira, automates data collection, and presents actionable insights to support Agile sprint planning. The tool successfully met these requirements by continuously updating sprint data, providing team-specific breakdowns, and incorporating feedback-driven improvements. Key enhancements, such as excluding irrelevant port tickets, refining the QA team’s work analysis, adding Criticality and Epic classifications, visualizing estimation accuracy, supporting late time logging, and improving estimation parsing, further tailored the tool to the team's needs. These iterations demonstrated the importance of continuous feedback and adaptability in software development.

While the tool has significantly improved the sprint review process, there are opportunities for further enhancement. Future developments could include more advanced analytics, such as trend analysis over multiple sprints, machine learning-based estimation predictions, or deeper integration with other Agile workflow tools. Additionally, expanding the tool's usability beyond the DSR team to other departments could further validate its effectiveness and provide broader organizational benefits.

Overall, this thesis contributed a practical, data-driven approach to sprint analysis, aligning Agile planning with real-time performance insights. The implementation of this tool not only streamlined sprint retrospectives but also fostered a culture of continuous improvement within the team. By providing accurate and automated feedback on estimation quality, the tool empowers teams to make more informed decisions, ultimately leading to better resource management and improved software delivery outcomes.

Acknowledgment of Generative AI Use

While writing this thesis, I utilized OpenAI's ChatGPT (November 2024 version) as a supportive tool for brainstorming and enhancing clarity in my text. Specifically, I prompted the model with general questions such as “What are common challenges in Agile sprint processes?” and “How can story point estimation accuracy be evaluated effectively?” The outputs provided insights that were critically evaluated and integrated into the research process. Additionally, ChatGPT was used to refine the phrasing of certain sections, helping to make the text more flowing and aesthetically polished. All content was extensively modified and contextualized to meet the academic standards and align with the specific goals of the thesis. The use of ChatGPT was supplementary and aimed at improving efficiency and focus during the drafting phase.

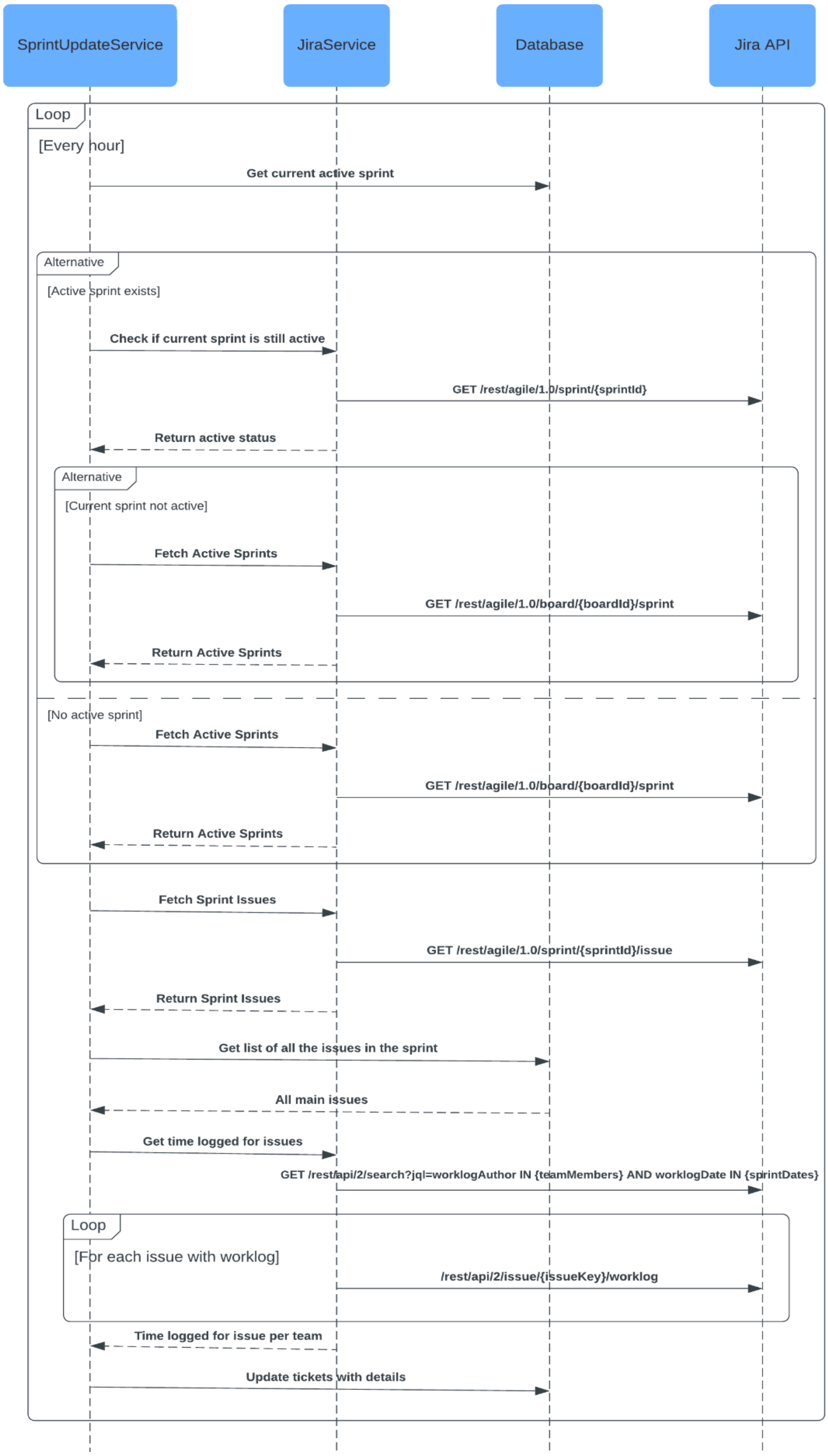
References

1. K. Beck, M. Beedle, A. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas. Manifesto for Agile Software Development. 2001.
https://agilemanifesto.org/. (07.10.2024)
2. Hoda, R., Salleh, N, Grundy, J. The Rise and Evolution of Agile Software Development IEEE Software, 2018, vol. 35, no. 5, pp. 58-63.
https://ieeexplore.ieee.org/document/8409911. (18.11.2024)
3. Matthies, C. Agile Process Improvement in Retrospectives. IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Montreal, QC, Canada, 2019, pp. 150-152.
https://ieeexplore.ieee.org/document/8802788. (18.11.2024)
4. Mallidi, R. K., Sharma, M. Study on Agile Story Point Estimation Techniques and Challenges. International Journal of Computer Applications, vol. 174, no. 13, Jan. 2021.
https://www.ijcaonline.org/archives/volume174/number13/mallidi-2021-ijca-921014.pdf. (13.12.2024)
5. Cohn, M. Agile Estimating and Planning. 2005.
6. Molokken, K., Jorgensen, M. A review of software surveys on software effort estimation. International Symposium on Empirical Software Engineering. 2003, pp. 223-230.
https://ieeexplore.ieee.org/document/1237981. (18.11.2024)
7. Zahraoui, H., Idrissi, M. A. J. Adjusting story points calculation in scrum effort & time estimation. 10th International Conference on Intelligent Systems: Theories and Applications, 2015, pp. 1-8.
https://ieeexplore.ieee.org/document/7358400. (18.11.2024)
8. Chongpakdee, P., Vatanawood, W. Estimating user story points using document fingerprints. 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2017, pp. 149-152.
https://ieeexplore.ieee.org/document/8342885. (13.12.2024)
9. Tanveer, B., Guzmán, L., Engel, U. M. Understanding and Improving Effort Estimation in Agile Software Development—An Industrial Case Study. IEEE/ACM International Conference on Software and System Processes, 2016, pp. 41-50.
https://ieeexplore.ieee.org/document/7831574. (18.11.2024)
10. Jorgensen, M. Contrasting ideal and realistic conditions as a means to improve judgment-based software development effort estimation. Information and Software Technology. V. 53, I. 12, 2011, p. 1382-1390, ISSN 0950-5849.
https://doi.org/10.1016/j.infsof.2011.07.001. (18.11.2024)
11. Vlaanderen, K. Jansen, S., Brinkkemper, S., Jaspers, E. The agile requirements refinery: Applying SCRUM principles to software product management. Information and Software Technology. vol. 53, i. 1, 2011, p. 58-70, ISSN 0950-5849.

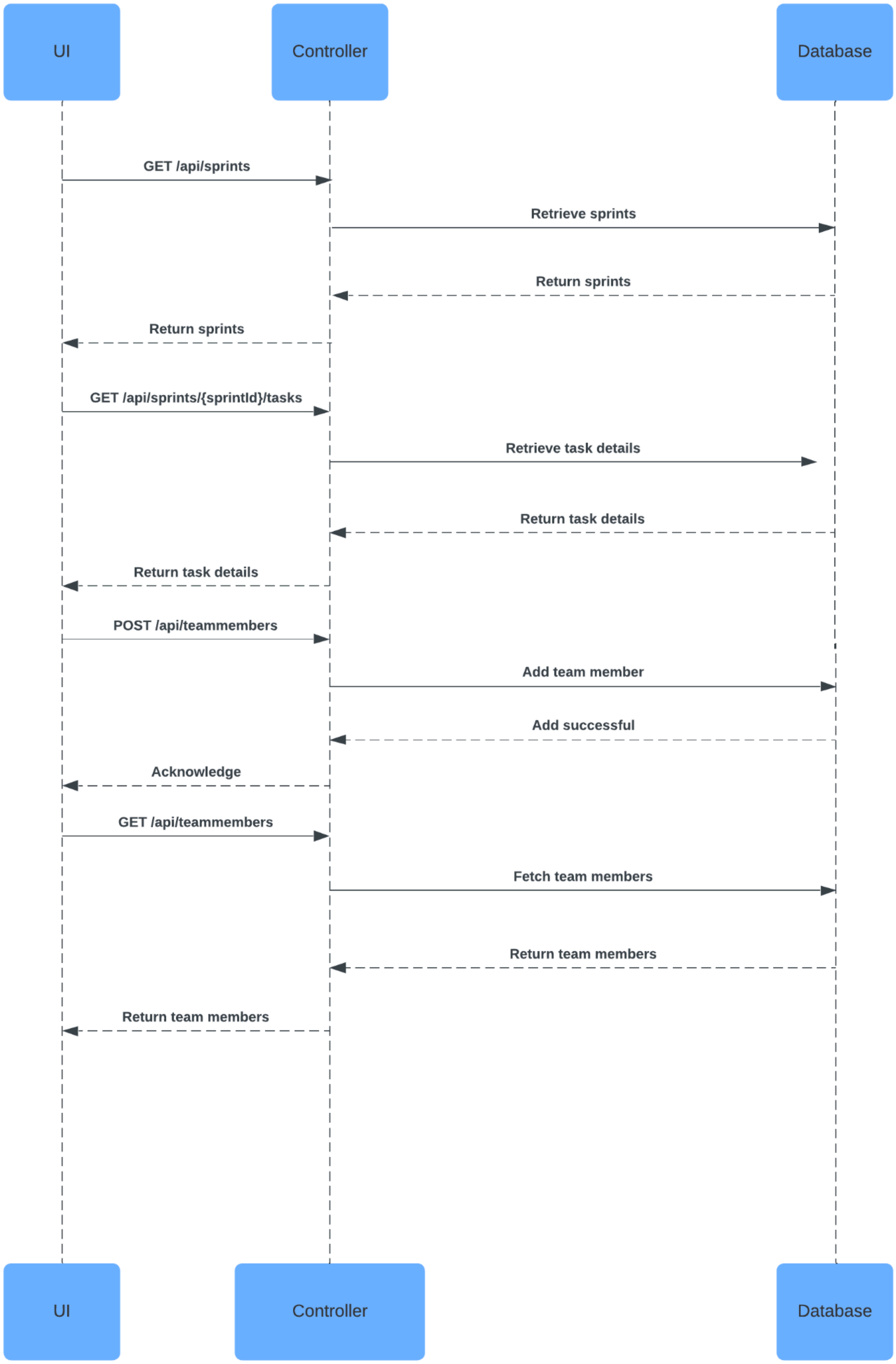
https://doi.org/10.1016/j.infsof.2010.08.004. (18.11.2024)
12. N. K. Sravan, V. A. Vuyyyru, P. Gottapu, A. Theeda, and L. Peddineni. Agile Management Tools: Technological Evaluations and Future Archetype. 2024 4th International Conference on Pervasive Computing and Social Networking, 2024, pp. 914-918, doi: 10.1109/ICPCSN62568.2024.00154. https://ieeexplore.ieee.org/document/10607751. (13.12.2024)
13. Pasuksmit, J., Jiang, F., Thornton, K., Friedman, A., Fuksmane, N., Kohout, I., Connor, J. Improving Agile Planning for Reliable Software Delivery. 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), Melbourne, Australia, 2023, pp. 25-26. https://ieeexplore.ieee.org/document/10174096. (13.12.2024)
14. Hundhausen, C., Conrad, P., Tariq, A., Pugal, S,. Flores B,. An Empirical Study of the Content and Quality of Sprint Retrospectives in Undergraduate Team Software Projects. Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '24). Association for Computing Machinery, New York, NY, USA, 2024, pp. 104–114. https://doi.org/10.1145/3639474.3640074. (08.04.2025)
15. Lenarduzzi, V., Fucci, D. Towards a Holistic Definition of Requirements Debt. 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Porto de Galinhas, Brazil, 2019, pp. 1-5. https://ieeexplore.ieee.org/document/8870159. (08.04.2025)
16. Aranda, J., Easterbrook, S. Anchoring and adjustment in software estimation. ACM Sigsoft Software Engineering n. 30, pp. 346-355. https://www.researchgate.net/publication/221560646_Anchoring_and_adjustment_in_software_estimation. (08.04.2025)
17. Life at Playtech Estonia. https://www.playtechpeople.com/country/estonia/. (25.03.2025)

Appendix

I. Sequence Diagram for Business Logic



II. Sequence Diagram for Controllers



I. **License**

Non-exclusive licence to reproduce the thesis and make the thesis public

I, Karl Soosalu ,
(author´s name)

1. grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the digital archives of the University of Tartu until the expiry of the term of copyright, my thesis

   Improving the sprint review and delivery process by enhancing the monitoring system ,
   (title of thesis)

   supervised by Dietmar Alfred Paul Kurt Pfahl ;
   (supervisor´s name)

2. grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright;
3. am aware of the fact that the author retains the rights specified in points 1 and 2;
4. confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Karl Soosalu

15/05/2025