Automated Jira Data Analysis for Optimised Project Supervision and Delay Detection

Johannes Koop

2020 Laurea

Laurea University of Applied Sciences

Automated Jira Data Analysis for Optimised Project Supervision and Delay Detection

Johannes Koop
Business Information Technology
Bachelor’s Thesis
June 20, 2020

Laurea University of Applied Sciences
Degree Programme in Business Information Technology
Bachelor of Business Administration

Abstract

Johannes Koop

Automated Jira Data Analysis for Optimised Project Supervision and Delay Detection

2020 Pages 67

CAS Software AG is a mid-sized Software company in Karlsruhe, Germany. They use Atlassian's Jira Project Management Tool to organise and monitor customer projects.

During a 5-month internship at CAS Software, I was able to analyse the management and developer side of a software development project using Jira.

Currently, Project Managers need to analyse data manually to successfully reallocate resources, optimise cost-profit-ratio or organise developer teams within a project. Investigating the delayed issues requires downloading of Jira Tempo reports and manual data entry in Excel tables. Therefore, this task is often neglected for its extensive effort.

The objective of this thesis project was to provide a solution to support Project Managers at CAS in supervising projects with the support of a Jira-ticket monitoring tool.

Interviews and surveys were chosen as methods to evaluate the underlying problem for ticket delays and current procedures. The so-called "TimeTrackingTool" is an already existing CAS-internal application that supports project controlling procedures by providing accumulative reports of Jira worklogs.

In the scope of the thesis project the TimeTrackingTool was extended by the "Delay Detection Mode". This mode analyses different metrics and provides the Project Manager with calculated estimates and delay predictions for individual Jira issues.

The Delay Detection Mode computes key performance indicators for Jira issues such as estimated time, aggregated time and time progress. With the support of the new mode, a Project Manager can prevent delays by reallocating developer resources, changing priorities or further actions.

The resulting software solution is used for a range of similar projects within CAS Software to avoid unnoticed delays for individual tickets and improve project performance overall. Previous calculation of delays and estimates were done in around 45 minutes. The Delay Detection Mode is providing results on average in less than 60 seconds.

Keywords: Jira, Project Management, Java, Time Management, Resource Management

Table of Contents

List of Abbreviation:

CAS	The client company CAS Software AG from Karlsruhe, Germany.
Confluence	A collaboration software program developed and published by Atlassian. Used by CAS for documentation.
Delay Detection Mode	Third mode added to the TimeTrackingTool as part of this thesis project.
Epic	Group of multiple Jira Issues. Used to organise Issues and form milestones for teams and map single issues to entire features.
Git branch	A separate parallel workspace to avoid working the main branch of the version control.
Git checkout	A command to change to a specific branch.
Git pull	A command to load latest changes from the version control server.
Git push a branch or commit	A command to upload local changes to the version control server.
Issue	Represents a task in a Jira project. An issue could represent a story, a bug or other technical tasks.
JavaScript Object Notation (JSON)	Open standard file format to transfer data in a human-understandable text.
Jira	Project Management software by Atlassian that supports bug tracking and agile project management.
JSON get-methods	Method to obtain certain elements of the JSON object.
SDLC	Software Development Life Cycle.
Sprint	Term of SCRUM: A fixed time period with a set amount of Issues for a team.
Story Points	A measurement to estimate the complexity of a task.

Tempo	Time Tracking extension for Jira.
TTT	CAS-internal “TimeTrackingTool” application.
Worklog	Users enter their task and the time spent on Jira issues in the form of entries called worklogs.

1. Introduction

In times of fierce competition, companies need to improve their productivity. Staying competitive can only be achieved by fighting many different battles. Every successful project will follow and balance the metrics of the project management triangle. These metrics define the triple constraints of quality, scope and time.

An innovative company can adopt new technologies to improve cost-control. Time-monitoring software supports the Project Manager to track the progress of a project, and investigate on delay root causes to find adequate counter-actions.

Software developing companies that focus on customer projects can have difficulties determining the exact cost of a project. This problem arises because of almost non-existent marginal cost and the difficulty to exactly know how much time was spent by developers on a software project. To mitigate this issue, a form of time tracking needs to be conducted to evaluate the time spent on a customer project by each Project Manager and developer.

Time Tracking itself is not new, emerging over 130 years ago. The Tabulating Machine Company is one early example, developing a tabulating machine to punch holes into a strip of paper to store information about employee work hours. (Cruz 2019)

1.1 Client company: CAS Software AG

CAS Software was founded in Karlsruhe, Germany by Martin Hubschneider and Ludwig Neer in 1986. CAS found early success developing a sales support and customer service system for Daimler, the parent company behind Mercedes-Benz.

CAS was successful with the help of a major client. Today these systems are still used by over 3000 Daimler employees in 13 different countries. The CAS Group has over 400 employees with a 40-million-euro annual turnover. CAS main product is the distribution of their relationship management platform genesisWorld. (CAS n.d.)

CAS Software AG is using the project management software Jira as a self-hosted on-premises solution. CAS Software is branding itself with the title “Software Made in Germany”. CAS is trying to avoid using foreign services for their business operations to maintain the title. Especially, since the US-court decision from August 2014 obliges US firms to reveal data to federal US institutions, even if the data lies on a foreign server. (CAS 2014)

1.2 Background: Jira by Atlassian

Atlassian is a software company founded in Sydney, Australia in 2002 by Mike Cannon-Brookes and Scott Farquhar. The agile project and issue tracker Jira is their first product, and most profitable to date. Over the last 18 years Atlassian has declined many investor and buyer offers. Instead Atlassian focused on acquiring more products and seamlessly integrating them into their continuously growing ecosystem. (Atlassian n.d.)

Jira is the market leader with a market share of around 32%. (Datanyze.com, n.d.)

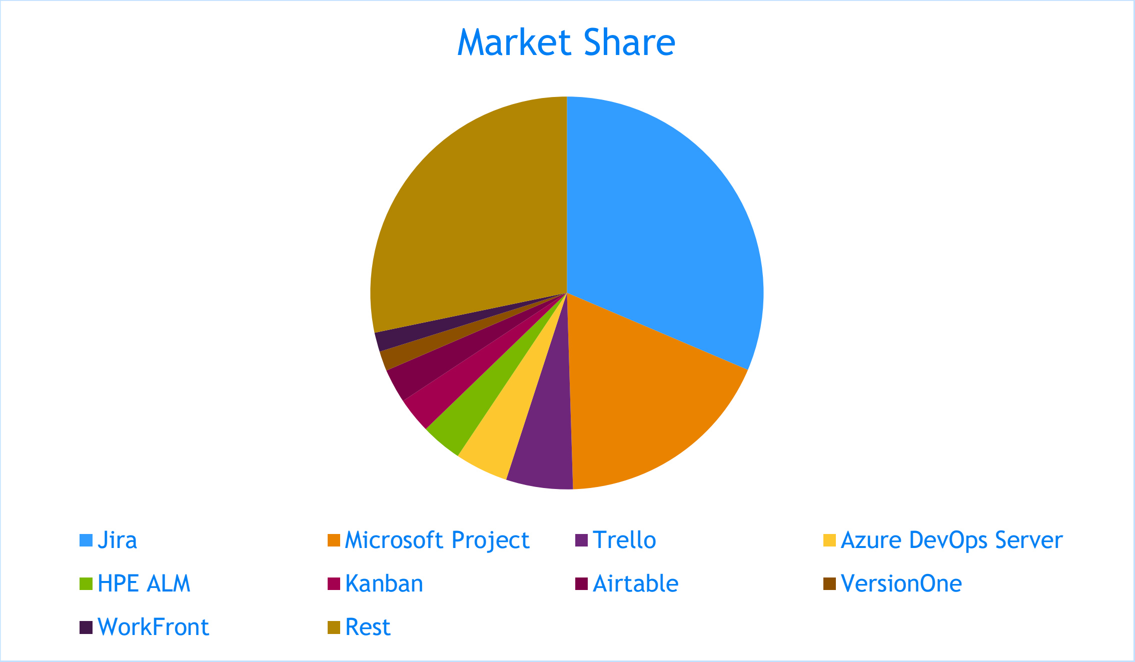
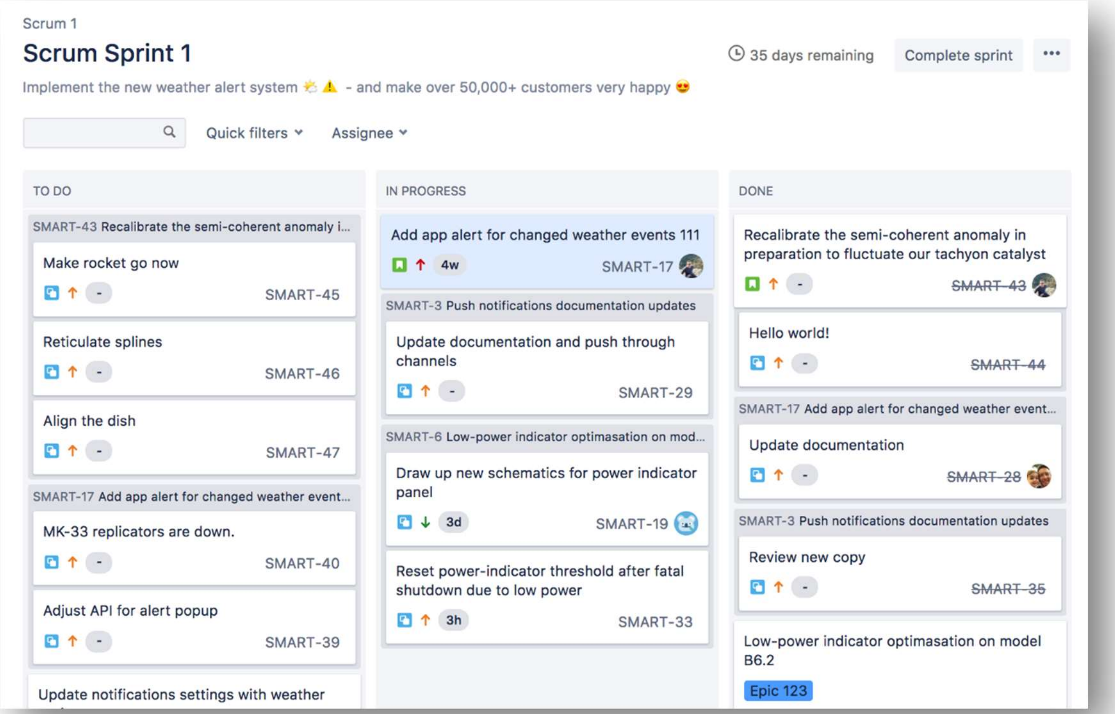


Table 1 - Market share project management tools

1.3 Business Needs

CAS is using Jira as their main tool to provide a platform for agile software projects. Jira issues are summarised on a Jira board (see Figure 1) in different columns to indicate their workflow status. For example, in Figure 1 the SMART-17 ticket is in the “In Progress” column. This indicates that a developer is working on this ticket. The SMART-44 ticket is in the “Done” column to indicate the work on this ticket has been completed.

Software development teams will organise their different sprints. A sprint is a defined time period for a Jira board. During one sprint there will be a defined amount of tickets on the board. The tickets will determine the developer work for the sprint.



Scrum 1

Scrum Sprint 1

Implement the new weather alert system ⛅ ⚠ - and make over 50,000+ customers very happy 😍

TO DO

SMART-43 Recalibrate the semi-coherent anomaly i...

Make rocket go now - SMART-45

Reticulate splines - SMART-46

Align the dish - SMART-47

SMART-17 Add app alert for changed weather event...

MK-33 replicators are down. - SMART-40

Adjust API for alert popup - SMART-39

Update notifications settings with weather

IN PROGRESS

Add app alert for changed weather events 111 4w SMART-17

SMART-3 Push notifications documentation updates

Update documentation and push through channels - SMART-29

SMART-6 Low-power indicator optimasation on mod...

Draw up new schematics for power indicator panel 3d SMART-19

Reset power-indicator threshold after fatal shutdown due to low power 3h SMART-33

DONE

Recalibrate the semi-coherent anomaly in preparation to fluctuate our tachyon catalyst - ~~SMART-43~~

Hello world! - ~~SMART-44~~

SMART-17 Add app alert for changed weather event...

Update documentation - ~~SMART-28~~

SMART-3 Push notifications documentation updates

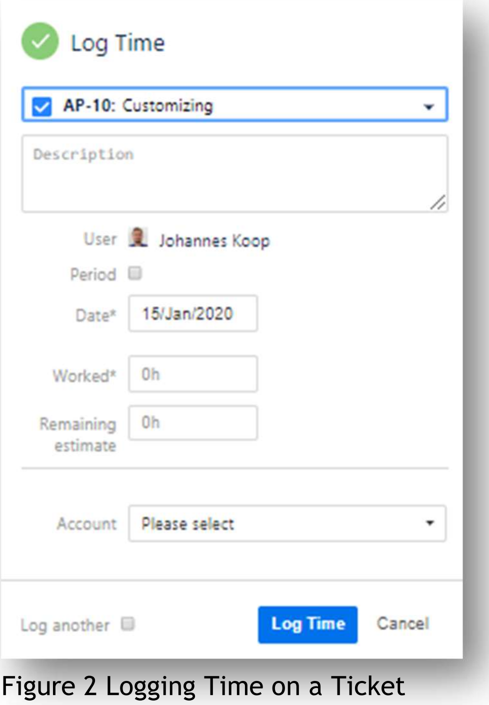
Review new copy - ~~SMART-35~~

Low-power indicator optimasation on model B6.2

Epic 123

Figure 1 Example Jira Board

CAS Software also uses Jira to track time for projects. Developers are logging the time they spent working on a ticket in the ticket worklog function. Jira's analysis of the logged times by the developers helps the Project Managers to create detailed reports about the current expenses on a project. These reports enable faster project cost supervision.



Log Time

☑ AP-10: Customizing

Description

User Johannes Koop

Period □

Date* 15/Jan/2020

Worked* 0h

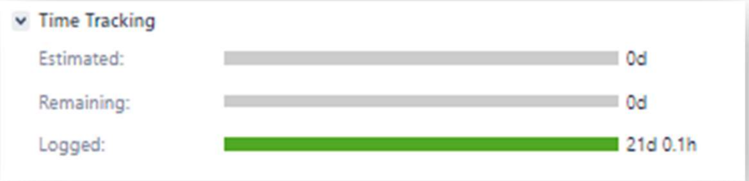
Remaining estimate 0h

Account Please select

Figure 2 Logging Time on a Ticket

In Figure 2 you can see the standard Jira menu to log work time on a ticket. You can choose the respective ticket, worked time and Account that will be charged with the cost.

In Figure 3 you can find the Time Tracking diagram. This diagram displays the logged time and estimated time for a ticket. The estimated time value needs to be filled manually. Not all teams are using this value in their project workflow. The missing estimated time can make it difficult to have a grasp on the current progress status of a ticket. For example, a ticket could have 10h logged, but no adequate progress was made.



Time Tracking

Estimated: 0d

Remaining: 0d

Logged: 21d 0.1h

Figure 3 Worklog Diagram

1.3.1 Estimation in Story Points

Some of the teams at CAS are estimating the complexity of a ticket in Story Points. Story Points do not have a set conversion in hours and minutes. Depending on the complexity of an issue and the individual skillset of a developer, the needed time for an issue can be roughly predicted. Project Managers can export Jira Reports with a calculated average time spent on a Story Point during a Sprint. This is one of the indicators to establish estimates during a project.

1.3.2 Detecting Delays

CAS Project Managers have tried to find a solution to simplify delay detection in one of the third-party plugins for Jira. So far there was no adequate approach to satisfy CAS' business needs. To support decision making CAS Project Managers need to look at the entire development and velocity of a project.

Detecting ticket delays could prevent delays for the entire project. Past experience has indicated that CAS developers do not always ask for help from co-workers when encountering difficulties. Despite the time constraints, developers often attempt to solve the difficulties independently. (Schrumpf 2019)

1.4 Objectives

The goal of this thesis is to improve existing project supervision workflows with a custom software solution. The software should save the Project Manager time when supervising Jira projects to allow more frequent and thorough delay detection. Before the project was started some research was done to understand the current situation. The research consists of interviews and a survey to analyse the current situation for developers and Project Managers. It was important to analyse the teamwork amongst CAS AppFactory developers to test how they behave towards each other, when problems in a project arise that can hinder the project progress. Delays can be mitigated, if developers receive the required help or other developers take over the task.

Furthermore, it was crucial to evaluate how Project Managers detect and mitigate delays. Project managers usual procedures will have an effect on the outcome of the TimeTrackingTool. If the current project controlling processes are sufficiently sophisticated, the improvement with the TimeTrackingTool might not be justified.

The project goal has two related research objectives.

The first research objective of this thesis is to understand the underlying project supervision processes of CAS Project Managers. Analysing these processes helps to provide a foundation to investigate on downfalls and potential improvements. This allows to define requirements of the thesis project. My previous work experience and surveying CAS Project Managers will provide the data to comprehend the obstacles of current project supervision procedures.

The second research objective is to investigate if there are personnel-related variables that contribute to overall project delays. Finding these variables could help to define more clear thesis project requirements and CAS Project Managers can use this information to adapt their project strategies for improved project success. The personnel-related variables will be evaluated with CAS developer surveys based on the Project Manager interview results. Project Managers can provide broad insight to suspected delay-contributing variables.

2. Theoretical Framework

2.1 Project Controlling Processes

According to the Project Management Body of Knowledge project management consists of five process groups:

- Initiating
- Planning
- Executing
- Monitoring and Controlling
- Closing

Throughout the monitoring and controlling phase the project manager tracks, reviews and regulates the progress and performance of a project to detect budget and time transgressions. To maintain the latest update on project progress and performance frequent key performance indicators calculations are required.

Continuous monitoring enables the Project Manager to identify project areas that require additional adjustment to guarantee the smooth operation of the project. For example, when a task exceeded its schedule and the responsible supervisor detected the transgression early enough it allows to adjust the resource allocations to prevent a chain reaction that will cause more delays over the entire project life time.

Controlling the constraints cost, scope and schedule is essential for a successful project. These constraints are described as the project management triangle. Changes to one of the constraints will require adjustments of the other constraints to avoid quality compromise. For example, extending the scope without increasing the schedule or cost will lead to quality losses. Unless more resources are used to cope with the higher workload, which increases cost.

According to the Project Management Body of Knowledge (PMBOK) the processes in the Monitoring and Controlling group are crucial to supervise the progress and performance of a project. The definition of the PMBOK is important in this study, as it assists to find weaknesses in the current processes of CAS Project Managers. (PMBOK2008)

Investigating the effect of time pressure on cognitive performance has been studied for several years. Project Managers try to maximise the productivity to increase overall project performance. However, studies found that excessive and prolonged time-pressure can have negative effects on the performance of employees. For the client company an individual research approach is needed to find variables that can contribute to performance decline. (Kramer et al. 2002)

2.2 Project Human Resources

Project Human Resource Management includes all processes that organise, manage and lead members of the project team. Team members have assigned roles and responsibilities within a project. The four main processes of project human resources are:

- Develop Human Resource Plan
- Acquire Project Team
- Develop Project Team
- Manage Project Team

The process of managing project teams comprises tracking team member performance, providing feedback and resolving issues. The project manager should observe team behaviour to examine the factors that affect project progress. Managing the project team requires to be aware of all human resource factors that might compromise team performance. These factors include environment, geographical location, team communication, cultural issues and other personnel-related factors. For example, digital collaboration is trending but in certain circumstances team work can be boosted by working together in person. Team communication and team work culture build have great impact in teams with unfamiliarity amongst team members. (PMBOK 2008)

The definition of managing project teams from the Project Management Body of Knowledge provides a foundation to evaluate the situation at CAS. This definition allows to compare the results of the study to an ideal value defined by the PMBOK.

3. Research Methodology

Investigating the effect of time pressure on cognitive performance has been studied for several years. Project Managers try to maximise the productivity to increase overall project performance. However, studies found that excessive and prolonged time-pressure can have negative effects on the performance of employees. For the client company an individual research approach is needed to find variables that can contribute to performance decline. (Kramer et al. 2002)

3.1 Qualitative

There have been two interviews with Project Manager of the AppFactory. They have been selected as participants because they will be the main users of the software project behind this bachelor's thesis. Both Project Managers have great insight in the business operations and can evaluate downfalls of workflows. The interviews have been done separately in a semi-structured manner. There were prepared questions to help guide the interview, but time was also allocated for participants to discuss the importance of other project aspects, if necessary. The interviews were both 30 minutes long and were documented in a Confluence page. The main points discussed in each interview were:

- Current project controlling procedures
- Using the TimeTrackingTool
  - Benefits
  - Downfalls
- Requirements for the Delay Detection Mode

3.2 Quantitative

The goal is to gather opinions from multiple software engineers in different locations. With an online survey the physical distance between the office in Germany and Hungary are irrelevant. All participants can fill out the anonymous survey at their own discretion.

The Google Forms survey was accessible from 28.11.2019 – 15.01.2020 and is comprised of eight non-suggestive questions to evaluate the work environment of software engineers in the AppFactory department of CAS. All employees of the CAS AppFactory have been selected as the target group for the survey. 50 people received the invitation e-Mail on the 28.11.2019 with a link to access the survey.

To ensure a bias-free outcome of the survey, the question should avoid leading the participants. For example, the question “Do CAS developers help each other?” is the non-leading approach. By adding the word “constantly” to the question it becomes suggestive and leading

“Do CAS developers help each other constantly?”. The participant is likely to answer negatively as developers are not constantly helping one another. (Bhat n.d.)

Five out of eight questions are based on rating scales. Based on the Likert scale, these questions allow the participants to provide streamlined and fast answers. The user does not need to write their own text. Furthermore, all participants of this survey are non-native English speakers. Questions that require answers with written text answers in a second language introduce the risk of misinterpreting answers. Rating scales avoid misinterpretation, because the results will be values from one to seven. Therefore, these values require little effort to compute mathematically and compare the responses of several participants, or examine correlations between answers. (LaMarca 2011)

The survey had a moderate response rate of 41.6%. Nine employees from Hungary and 16 from Germany have filled out the questionnaire. The response rate may be low because the invitation email was sent to all AppFactory employees, including non-developers. The email explained that this survey is for participants with developer experience exclusively.

4. Research Outcome

The key findings of the conducted research were used to determine the approach for the underlying software project and to answer the research question. The outcome of the interviews and survey was evaluated individually, but contributed to the development of the project as a whole.

4.1 Project Managers

Steffen Euch and Marcel Schrumpf are both team leaders and Project Managers in the AppFacotry unit of CAS. Steffen is a user of the current TimeTrackingTool and knows its functionalities. Steffen and Marcel are the main stakeholders and primary users of this thesis project's outcome. They have set the requirements and limitations for the TimeTrackingTool software based on their experience in project controlling and project supervision.

4.1.1 Interview with Steffen Euch

The interview with Steffen Euch has helped to understand the current downfalls of project management procedures. Steffen is a primary user of the TimeTrackingTool. The TimeTrackingTool is an application, that helps Project Managers to create project controlling reports. To increase the usefulness of this tool for Steffen, the TimeTrackingTool should allow Steffen to access information about the current performance of Jira issues. The current standard export tools by Jira and the Tempo extension do not provide sufficient data to analyse the amount of worklogs in a specified timeframe. This makes cost controlling in a project more time consuming. As a Project Manager, Steffen is responsible to submit project performance reports to stakeholders. Having faster access to reports that analyse the time spent and time planned would eliminate manual reporting for Steffen and save time. (Euch 2019)

4.1.2 Interview with Marcel Schrumpf

Marcel Schrumpf has given further insight about the problems Project Managers encounter during project supervision. According to Marcel's interview, he is facing challenges when trying to predict delays and supervise delayed issues. Currently delays are often detected towards the end of a sprint. Project Managers need to manually evaluate Jira Tempo export to investigate if a Jira issue is delayed. The extensiveness of this task leads to its neglect. (Schrumpf 2019)

Usually, the Project Manager will update the project plan towards the end of a sprint. Updating the project plan exclusively at the end of sprints disables the Project Manager to react to delays throughout the sprint. (Schrumpf 2019)

Detecting delays is time consuming, but establishing predictions is time intensive as well. To create predictions for issues, the Project Manager must calculate average times in project. As values in project constantly change, so must the prediction. Maintaining an up to date prediction would require constant updating. This leads to the neglect of prediction calculations. (Schrumpf 2019)

Marcel is currently not a user of the TimeTrackingTool, but to improve the usefulness of the TimeTrackingTool the application should calculate predictions for each issue in a project based on continuously updated data. The tool should allow the user to obtain instant information about the performance of a project and its issues. (Schrumpf 2019)

4.2 Developers

The first survey questions were asking the participant about their location, seniority and experience level. These questions were trying to prove any correlation to team work culture.

This questionnaire aimed to prove, if AppFactory developers cannot solve problems themselves whether they request assistance from other team members. If they help each other when encountering problems, delays within the project can be prevented. But if the developers are not sharing their difficulties, it could lead to overall project delays.

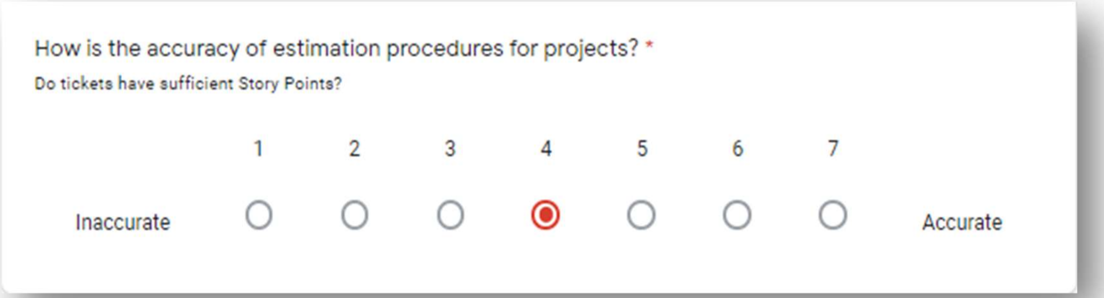
Appendix 1 depicts the ratio between Hungarian and German developers at CAS. Nine developers from Szeged and 16 from Karlsruhe participated in the survey. The average participant has worked 5,9 years for CAS and has 6,96 years of professional developer experience. The survey’s main questions were regarding teamwork. They depict the satisfaction level of the developers for working with each other on team projects.

The vertical axis of the chart in Appendix 10 displays the answer scale. 0 is equivalent to “No, I don’t” or “Difficult”. 7 is the opposite of 0, meaning “Yes, I do” or “Easy”. The horizontal axis represents each participant of the survey.

By analysing this chart, we see that the average developer at the CAS AppFactory unit is likely to help other developers. Furthermore, AppFactory developers believe that asking for help amongst each other is easy. The overall positive result of the teamwork questions indicates that developers are satisfied with teamwork amongst their peers. This is beneficial for project teamwork.

The results demonstrate that it is likely for developers to receive help to solve an issue and avoid unnecessary delays. Nevertheless, it is not improbable that technical difficulties arise and the developer is spending disproportionate amounts of time on the issue. Also, individual developers might not seek help to save face. Furthermore, developers might not alarm the Project Manager, in case of difficulties, and will try to solve the issue with help from peers. This disables the Project Manager from ordering external help early on, and from including delays in project planning

.



How is the accuracy of estimation procedures for projects? *

Do tickets have sufficient Story Points?

	1	2	3	4	5	6	7	
Inaccurate	○	○	○	●	○	○	○	Accurate

Figure 4 - Google Forms Question

Additionally, the question relating to the accuracy of estimation procedures produced the lowest result out of the rating scale questions. The average for this question was 4.56. This can be an indicator for Project Managers, that CAS software developers in the AppFactory unit are not fully satisfied with the procedures or the outcome for estimating story points within a project. If estimations are lower than the relative work required, developers will feel time pressure. In a light form, time pressure can increase productivity and intrinsic motivation. However, if inaccurate estimations cause consistent time pressure for developers, this could lead to overall dissatisfaction and block creative cognitive functions. (Kramer et al. 2002)

Ultimately, there was no significant correlation between location, seniority, experience level, and team work culture amongst CAS developers.

4.3 Limitations

Interviewing Project Managers and conducting surveys with developers has helped to gather information to answer the research question. The research had following limitations.

4.3.1 Sample Size

With only 25 participants the sample size of the conducted quantitative research was relatively small. On the other hand, the invitation to this online survey was sent to all 50 developers of the AppFactory department. However, the small size of the AppFactory, the sample size is in relation to the number of employees relatively high. This leads to a participation rate of 41.6%.

The choice of participants and sample size limits the result of this survey to the AppFactory developers exclusively. The changes made to the TimeTrackingTool will only affect developers within the AppFactory, since Project Managers in the AppFactory are using this tool exclusively. Considering other developers for this survey would dilute the results.

4.3.2 Previous studies

There have been no previous empirical studies to document developer data within the AppFactory department of CAS. This required an entirely new approach to gather information about the Project Managers and developers.

4.3.3 Mitigation

To improve the accuracy of the small sample size of the survey, the focus was set on a specific target group. The invitation to the survey was only sent to AppFactory members. AppFactory developers will be the only units affected by the changes made to project controlling procedures. Furthermore, the quantitative study is complemented by the qualitative study. The interviews with Steffen Euch and Marcel Schrumpf helped to mitigate the effect of a small sample size by providing facts from previous projects. The conclusion was made by outweighing the input from the interviews and the interpretation of the survey.

5. Workflow

The workflow was established based on information from my personal experience as a project management intern, and the interviews with Marcel Schrumpf and Steffen Euch. Elaborating the workflow helps to understand the benefits of the TimeTrackingTool improvements.

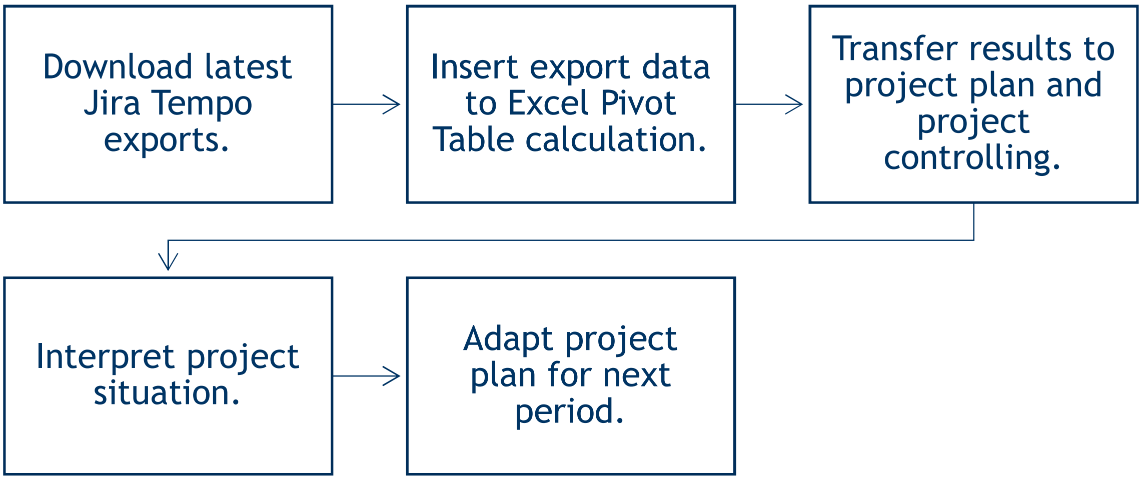

Download latest Jira Tempo exports.
Insert export data to Excel Pivot Table calculation.
Transfer results to project plan and project controlling.
Interpret project situation.
Adapt project plan for next period.

Figure 5 - Workflow Project Plan

5.1 Jira Tempo Procedures

Tempo is a plugin for Atlassian’s Jira project management tool. Tempo provides an array of time tracking tools and reports to enhance project controlling. Currently, CAS Project Managers are using Tempo’s Logged Time reports to accumulate project worklogs in a certain timeframe.

Appendix 12 depicts the export of a Logged Time report. The grid view of the report can be set in days, weeks, quarters or years. In the report example the selected timeframe was 01.03 – 31.03.2020 and set on a weekly interval. Project Managers will adjust this report to their requested project, timeframe and interval. (CAS - Jira n.d.)

When the desired variables have been set, an export of the report can be downloaded. The easiest way to further use the report data is in the form of a CSV or Excel file export. These reports will contain essential Jira issue data. However, the powerful feature of this logged time report is the ability to calculate the amount of worklogs for a specified time and project. Out of the box Jira cannot calculate worklogs for a certain timeframe. Without the Tempo plugin a Project Manager was required to accumulate worklogs by hand. Jira displays individual time logs and the overall total, but not the total of selected time frames. (CAS - Jira, n.d.)

Appendix 13 demonstrates an example Excel export from a Logged Time report. This report contains the entire cost of this example project for all of March. Project managers will use this information for further calculations in a different Excel file to include it in their project plan and project controlling. (CAS - Jira n.d.)

The data that comes from the export, will be used in project plans and other forms of final data processing. The resultant information obtained from project plans can help Project Managers to detect delays. For example, in Appendix 14 column AB, displays the ratio between planned project days and spent project days. If the ratio is higher than 100%, the issue has spent more time than planned. If the ratio exceeded 100% and the issue is not resolved yet, the Project Manager should investigate the reasons for this delay and react to them. However, the greatest challenge is the post-sprint investigation of delays. (CAS - Jira n.d.)

5.2 Downfalls

In most cases the delay investigation is only conducted at the end of the month. This investigation is extensive and time consuming. On average the update of the project plan and project controlling required around 40 minutes of work. As a consequence, this investigation is not done throughout the month to monitor the progress of a project. Also, this hinders the Project Manager from counter-acting to occurring delays during the month.

To detect delays in a Jira project contains copy and paste steps, which can lead to errors. When a Jira Tempo report was exported as an Excel or CSV file, the Project Manager must transfer it to another Excel table for further calculations. The step of manually copying data between Excel files can lead to errors. Carelessness errors may lead to pasting in a wrong column, which could lead to wrong results.

Furthermore, developers can correct their logged time in a project retrospectively. Retrospective correction of worklogs will require the Project Manager to update the project report data frequently. Project managers tend to update the project controlling on a monthly basis. During this update the last month’s worklog will be regarded exclusively. For instance, at the end of April 2020 a Project Manager will obtain the latest worklog list for the entire month, but a developer has corrected worklogs at the end of March 2020. If the Project Manager does not obtain all worklogs of a project, previous worklogs will not be visible. (Schrumpf 2019)

According to the 1-10-100-Rule, the cost increases ten-fold from prevention to correction and again to failure.

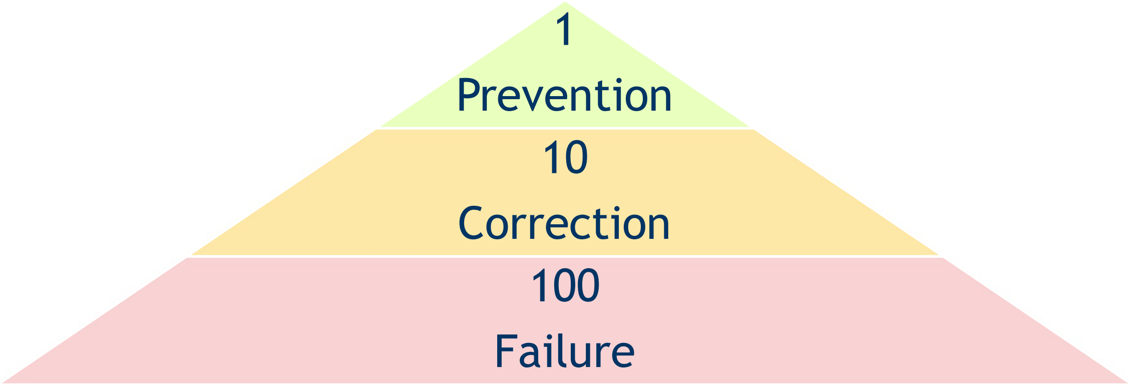


Figure 6 - 1-10-100 rule

As an example, to prevent errors in project controlling the Project Manager must ensure that the entered data, and therefore resulting reports, are correct. Preventing mistakes in project controlling could cost the Project Manager 20 minutes. The Project Manager can prevent errors by sanity checking the variables for the Jira Tempo report export, and paying extra attention when entering data. If the Project Manager does not prevent the error the correction will cost ten hours. After a project controlling report has been finalised it will be presented to stakeholder. Errors might be noticed now by stakeholders if the result are not conclusive. This will require the Project Manager to spend around 3 hours to remediate the error. Remediation is more time consuming because the error cause is often not instantly visible. The remediator will need to dedicate time to find the root cause of the errors. Finding wrong data entries in extensive Excel table calculations can be very time consuming, because of the number of cells, data and calculations which need to be assessed. If prevention and correction of errors has not prevented any errors, it will lead to false interpretation of project controlling reports. According to the interpretation of project controlling, project resources will be allocated. As an example, a task is within budget according to the false project controlling, but in reality, this task is over budget. This leads to the Project Manager not reallocating resources to mitigate the effect of the overbudget task on the project time line. Ultimately, the failure to prevent the error and to correct the error lead to a chain of events, that caused overspending and delays, which cost the entire company the equivalent of 30 work hours. (Syed 2009)

5.3 Mitigation of Downfalls with Delay Detection Mode

The downfalls of the current project controlling can be mitigated by developing a new mode in the TimeTrackingTool to suit the needs of Project Managers in the AppFactory. The development of this mode is discussed in chapter 6.

The Delay Detection Mode will allow the user to access instantaneous results to discover delays. Instead of manually obtaining the latest Jira exports and investigating the project situation, the Delay Detection Mode will provide all necessary information. This information provides the evidence to evaluate if counter measures are necessary to avoid delays. Furthermore, the low effort of using the Delay Detection Mode enables the Project Manager to use it throughout the month and increase the frequency of delay investigations.

The accuracy of report results can be improved with automated data imports. The tool will eliminate carelessness errors by reducing the amount of input variables required by the user. The user will only need to login with their Jira credentials and select the required project key. The user will not be required to copy and paste any information to obtain an update on the delays in a Jira project. The result table will display information about the performance of each Jira issue, which the Project Manager can use for his project resource allocations.

The tool will access the entire history of a Jira project and obtain all worklogs. This removes the danger of a Project Manager not refreshing previous project controlling data, when developers have updated previous worklogs. The Project Manager will not be required to reassure the validity of project data in project controlling reports.

The Project Manager can focus on allocating resources and making management decisions based on the results of the tool. The risk of allocating resources incorrectly is minimised by providing latest project controlling data, and results that are not prone to human carelessness errors.

After including the mitigation of downfalls with the Delay Detection Mode, the project management processes will be less extensive and more time saving. The updated workflow is faster and less extensive.

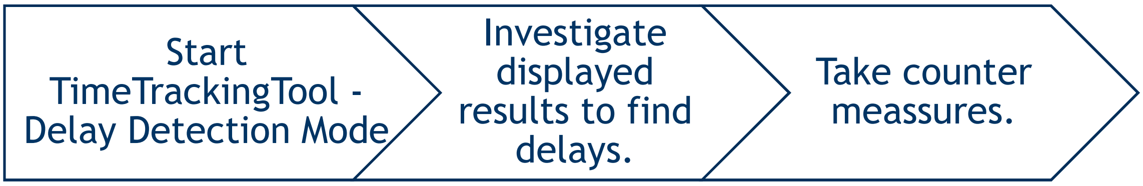

Start TimeTrackingTool - Delay Detection Mode
Investigate displayed results to find delays.
Take counter meassures.

Figure 7 - Updated Delay Detection Workflow

6. Development

To improve the understanding of the reader the existing TimeTrackingTool and the Delay Detection Mode must be discussed. The Delay Detection Mode is an added mode to the TimeTrackingTool.

6.1 Existing custom Jira Tools for CAS

The TimeTrackingTool is a JavaFX application. JavaFX is a set of graphics and media packages to allow developers to create desktop applications. The TimeTrackingTool code logic and graphical user interface is based on JavaFX 12.0.2.

Maven 4.0.0 is used a build automation tool and dependency management. Maven enables to run its commands to build and package the project. These are necessary to create an executable file of the project. The user is not required to open the project in an integrated development environment like IntelliJ or Eclipse. Instead the user can open the executable file to start the TimeTrackingTool.

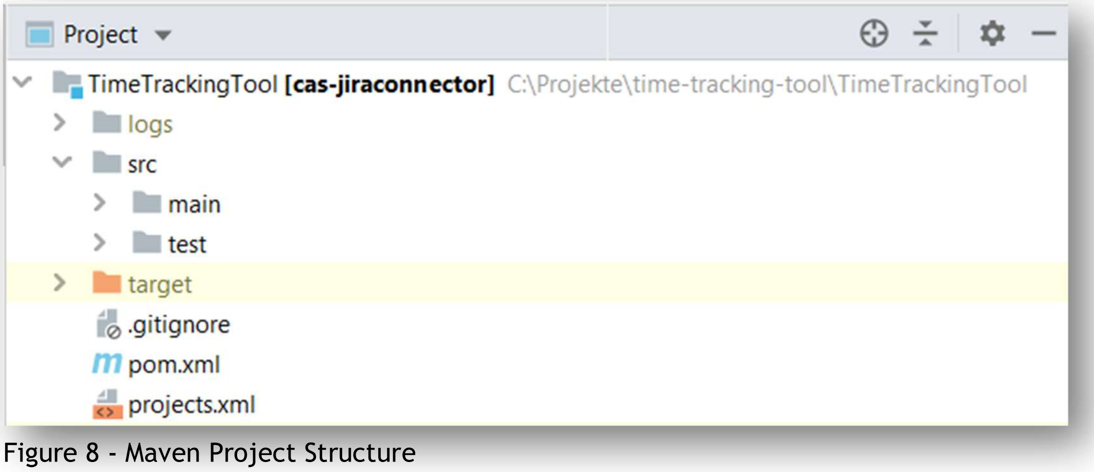

Project
TimeTrackingTool [cas-jiraconnector] C:\Projekte\time-tracking-tool\TimeTrackingTool
logs
src
main
test
target
.gitignore
pom.xml
projects.xml

Figure 8 - Maven Project Structure

Figure 8 depicts the classic structure of a Maven project. The src folder contains the logic components of the project. The main folder consists of the main functionality and logic of the application. Unit-test are located in the test folder.

The target folder is made up of unit-test outputs and execution variables. Each time compiling the application unit-tests are running and the results can be found here. All setting files are located in this folder. The timetrackingtool.yml lists variables for the Effort Mode that define hourly wages and save student settings. The wecloud.properties file contains location of the Excel file inlcuding all projects on the WeCloud storage system.

For Maven the pom.xml (Project Object Model) is fundamental. It contains all configurations for the project like dependencies, build directory and plugins. Maven reads the pom.xml and locates the directories of the main, test and target.

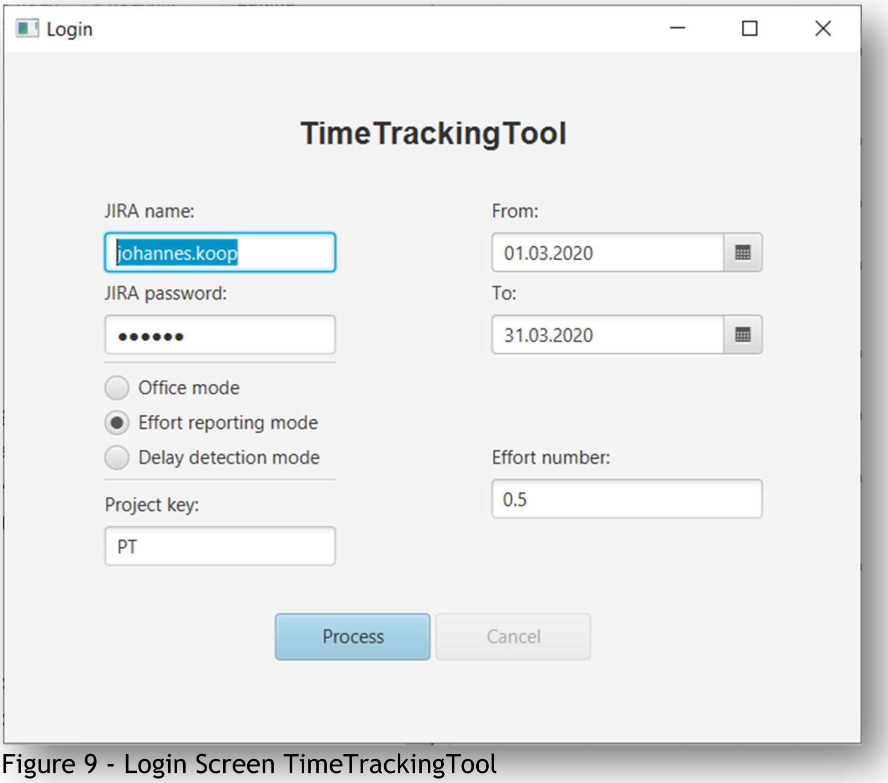
This JavaFX application obtains JSON objects from the CAS Jira servers using HttpGet requests. Http server request must be used for the TimeTrackingTool because the CAS Jira server does not support the Jira REST API.

The list of projects for the Office Mode are obtained from CAS own file storage service WeCloud. On the WeCloud lies an Excel list that contains all projects that will be evaluated in the Office Mode. The Office Mode is by CAS' branch in Hungary to create their monthly billing. The CAS Jira servers do not support the Jira REST API. The JSON objects contain all relevant information of worklogs, issues and projects. The JSON objects are mapped to their respective classes and variables for further use in the application.

Git was used as the version control system to track the commit changes of the software project throughout its development. The TimeTrackingTool project lies on a CAS hosted Gerrit server.

As a Project Manager Steffen Euch is using the TimeTrackingTool. The TimeTrackingTool consists of multiple modes. The effort mode helps by differentiating student and full-time employee cost. The effort mode allows to mark a user as a student and recalculate their productivity and cost according to the hourly rate of students. Also, the tool allows to select a timeframe, which enables the user to get precise information for monthly controlling reports. He is using the effort mode of the TimeTrackingTool to get cumulative worklog data for controlling reports. Appendix 16 depicts the result table of the effort mode. The effort mode is used to get a better overview of a project's performance in a given timeframe.

The effort mode requires the user to enter Jira login credentials, project key, effort number and timeframe. The effort number is used to differentiate the productivity between students and full-time employees. Productivity and hourly wages between students and full-time employees indicate a significant gap which needs to be represented in project controlling. Per hour, a student costs the company less than a full-time employee. Project keys are composed of at least two letters that are used to identify one Jira project. Within a company workspace every project key is unique. A project key is also used as a prefix for issues and epics.



TimeTrackingTool

JIRA name: johannes.koop

JIRA password: ●●●●●●

○ Office mode
● Effort reporting mode
○ Delay detection mode

Project key: PT

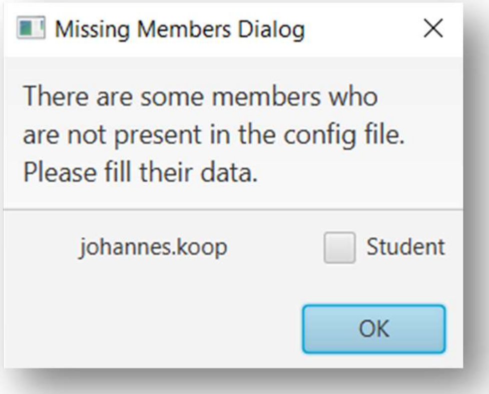
From: 01.03.2020

To: 31.03.2020

Effort number: 0.5

Figure 9 - Login Screen TimeTrackingTool

Using the effort mode of the TimeTrackingTool in comparison to standard Jira report comes with several benefits. In Figure 10 the TimeTrackingTool prompts the user to enter whether the project participant is a student. This has an effect on the displayed effective hours and cost of the worklog.



There are some members who are not present in the config file. Please fill their data.

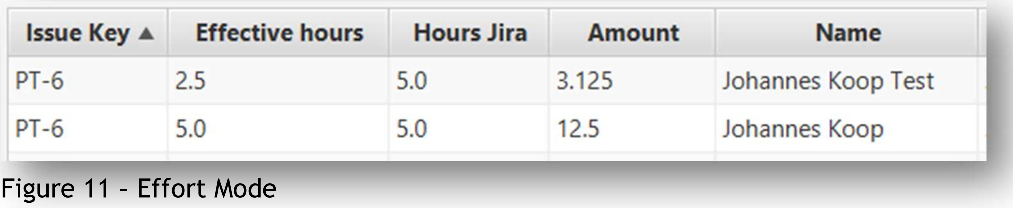
johannes.koop □ Student

Figure 10 - Student Check Dialogue

In Figure 11 the difference between students and full-time employees is indicated. The “Johannes Koop Test” account was selected as a student. Both accounts have logged five hours on the Jira issue PT-6. For the student account, the effective hours are halved according to the entered effort number. In this example the assumption is that a student is only half as productive as a full-time employee within one hour. That’s why they enter the effort number 0,5, which is the equivalent of 50% productivity for students.

In this example the daily rate for full-time employees is 20 Euros per day and for students 10 Euros per day. This gives us the values in the amount column:

1. Full-Time employee: 5 effective hours * 2,5 €/h = 12,5 €
2. Student: 2,5 effective hours * 1,25€/h = 3,125 €

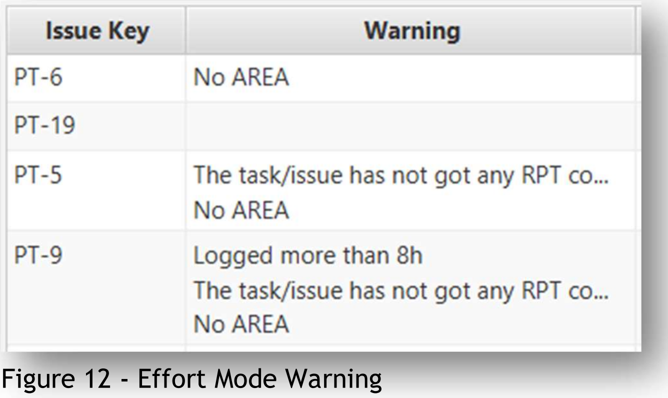


Issue Key ▲	Effective hours	Hours Jira	Amount	Name
PT-6	2.5	5.0	3.125	Johannes Koop Test
PT-6	5.0	5.0	12.5	Johannes Koop

Figure 11 – Effort Mode

The Project Manager can benefit from clear differentiation between students and full-time employees. Project controlling will be more accurate when the cost and productivity of students is taken into consideration. Normally a Project Manager would need to calculate student cost manually. There was no calculation from the Project Manager required to get the amount spent on project work by students and full-time employees.

In Figure 12, the warning column illustrates the warnings that can help the Project Manager to supervise Jira issues. These warnings are not depicted by Jira at any time, because they are not mandatory fields. However, these warnings have been set up to help the Project Manager maintain homogeneity amongst all worklogs. These warnings include missing epics, areas, components, or worklogs larger than 8 hours. Epics, areas and components help to categorise Jira issues into groups that allow users to find a set of Jira issues that belongs to a component or epic group. Project controlling can be done more precisely if worklogs are categorised by epics and components of a project. For that reason, project controlling will depict which epics or components were out of budget or caused delays.



Issue Key	Warning
PT-6	No AREA
PT-19	
PT-5	The task/issue has not got any RPT co... No AREA
PT-9	Logged more than 8h The task/issue has not got any RPT co... No AREA

Figure 12 - Effort Mode Warning

6.2 Project Management Approach

The analysis period has commenced in November 2019. Marcel will be one of the primary users of the finished software. As an experienced Project Manager, Marcel has worked with several Jira projects in the past. Based on his experience, Marcel can define the requirements which need to be met by this software project to fulfil his needs.

Throughout the first week of planning and analysing the requirements, Marcel and I have chosen a hybrid project management approach. Neither classic or agile project management delivered the guidelines and flexibility required. The classic project management required detailed planning for the entire life time of the project, and the agile approach could have caused too much overhead for a one-person team. (West n. d.)

During my internship as a software developer, with Marcel Schrumpf as my Project Manager, I have learnt to work within a SCRUM team. Agile project management is used by the majority of teams at CAS. (Schrumpf 2019)

The beginning of the thesis project has been conducted in a classic project management approach. During the initiation phase of the thesis, a project plan and project time line was created. These elements require the student to define the project scope from the start of their thesis project. The project plan and time line contain detailed information about the time frame and the milestones. Both documents specify the project elements as a waterfall scheme, because of the set dates and hierarchically order of implementation.

Agile Project Management	Classic Project Management
- Dailies with Marcel Schrumpf	- Milestone plan
- Weeklies with Oswaldo Montenegro	- Project plan
- Product Owner: Marcel	- Project scope
- Incremental software development with feedback cycles.	- Sequential project phases
- Agile mindset: What is most valuable for my client?	

Table 2 - Agile and classic project management elements

The combination of CAS leaning towards agile project management and the initialisation of the project by providing project plans and time lines to Laurea, allowed to start this thesis project in a hybrid project management approach.

In this thesis project the hybrid project management planning approach allowed to adjust requirements throughout the project life time if needed. Not all requirements were clear at the beginning of the project. Several times the requirements have been adjusted due to time constraints or inclusion of new needs. A solid work breakdown structure would have not allowed to change the requirements throughout the project. The execution and delivery are part of the agile methodology. Therefore, daily meetings with Marcel were used for status reporting. And during weekly meetings I have reported the development status to CAS software developer Oswaldo Montenegro. Oswaldo acted as a mentor for the code. I have created a milestone list to break down the project into manageable pieces, that allowed Marcel and I to break down the parts of the project and track the development progress.

6.3 Stages of Development

To clarify the progress of the Delay Mode the Delay Mode development has been separated into five stages. The stages are listed according to classic software development lifecycle. However, as part of a hybrid project management approach some stages were iterated multiple times.

6.3.1 Analysis

During the fundamental stage of Software Development Life Cycles, the conducted research helped to define the needs of the end users. During my time as an intern I helped Marcel to create project plans and monthly controlling reports. The combination of detailed interviews with Project Managers and my own experience were crucial for defining the software requirements. The most urgent need of Steffen and Marcel was to simplify the delay detection. Finding delays in a Jira project can be time consuming, because Jira project data needs to be exported and calculated in separate Excel sheets. Project managers have neglected delay detection throughout sprints, because of its effort.

6.3.2 Design

The existing TimeTrackingTool effort mode has four different tabs to group Jira issues:

1. user/components,
2. location/components,
3. epic, and
4. issue.

The issue tab displays one worklog per row and provides issue relevant information. (Appendix 17) After the analysis phase of the software project, it was apparent, that the effort mode's issue tab is a suitable base for the software project. Marcel and Steffen's requirements made clear that a delay detection on issue granularity is ideal instead of worklog granularity like the effort mode. Since the goal was to calculate and detect delays, the new mode for the TimeTrackingTool as part of the thesis will be named "Delay Detection Mode".

MockFlow is an online platform to create wireframes. With MockFlow a wireframe to outline the result of the thesis project was created. This wireframe has helped to envision the product owner Marcel and other stakeholders the result and to simplify the refining of the requirements. In case of unclear requirements or outcome for the software project, the wireframe has helped to visualise the goal. At the beginning of the project the wireframe serves as a form of visualisation, when the software itself is not ready. (Appendix 15)

With the effort mode issue tab as a base some columns were removed and some were added during the design process. These are the columns that were either reused, discarded or newly created:

Columns to be reused:	Columns to be discarded:	Columns to be added:
1. Issue Key	1. Working Description	1. Issue Status
2. Issue Summary	2. Effective Hours	2. Story Points
3. All Components	3. Hours	3. Sprint
4. Issue Type	4. Work Date	4. Assignee
5. Parent Key	5. Location	5. Aggregated Time Spent
6. Epic Link	6. Name	6. Estimated Time
7. Epic Name	7. Amount	7. Time Progress in %
8. Project Key	8. Area	
	9. Warning	

Table 3 - Delay mode columns

The discarded columns did not provide any added value to the goal of the software project or were not relevant. For example, all of the discarded columns, with the exception of area and warning, are not suitable for the new Delay Detection Mode, because these columns represent values that are attached to worklogs. A worklog contains information about the person logging work, the number of hours and the work date. A Jira issue does not contain that information.

The new columns were required for the goal of the Delay Detection Mode. The essential component of the Delay Detection Mode will be the calculation of average time spent per story point. Story points are a widely used estimation tool at CAS and other companies. Story points represent a relative amount of work. Using story points and the worklog sum of resolved issues allows to calculate the average time per story point. Only worklogs and story points from resolved issues are used in this calculation, because open issues would dilute the accuracy of the average time. At the beginning of an issue it has only a few worklogs, but the number of story points will stay the same. To avoid changing variables dilute the results only story points and worklogs from resolved issues will be used. This is the simplified calculation behind the Delay Detection Mode.

$$\boxed{Average\ Hours\ per\ Story\ Point\ {}^{h}/_{SP} = \frac{(\,Aggregated\ Time\ Spent\ of\ all\ resolved\ Issues)\ h}{(\,Aggregated\ Story\ Points\ of\ all\ resolved\ Issues)\ SP}}$$

Equation 1 - Average hours per story point

The result of the average time per story point is passed on to the next calculation. The average time per story point is multiplied with the amount of story points per issue. The result of this is the estimated time per issue in hours. The estimated time predicts the total of worklogs on an issue according to the average hours per story point. The automatic conversion of story points to an estimate of hours is a highly sought-after feature. (Schrumpf 2019)

The estimation procedure in software projects can be difficult due to the number of uncertain variables. Requirements are not always completely clear to all stakeholders and are likely to change throughout the project. The nature of software project requires them to work with foreign, continuously evolving technologies. With the automated conversion of story points to estimated hours, Project Managers can have more realistic estimates that help to deliver project success. (Radigan n.d.)

$$\boxed{\begin{aligned}&Estimated\ time\ per\ issue\ in\ h\\&\qquad = (Story\ points\ per\ issue)SP\ x\ (Average\ hours\ per\ story\ point)\ {}^{h}/_{SP}\end{aligned}}$$

Equation 2 - Estimated time per issue in h

To visualise and compare the current performance of each issue, the progress between aggregated time spent and estimated time needs to be considered. Calculating the quotient time progress per issue is the result of the division of aggregated time spent and the estimated time. Time progress per issue is the relative amount of actual worklog hours in comparison to the estimated time hours. The lower the time progress the less time has been spent with actual work compared to the estimate. If the time progress is higher than 100, the aggregated time spent exceeded the estimated time. This key performance indicator has to be used conservatively as the variables included in the calculation are changing throughout the project life time. Especially, during the early stage of a project variable fluctuations have a strong effect on the average time per story point. Time progress only represents the current situation and does not reflect the development overt time. For example, the time progress of an issue can be at 10% and two weeks later the time progress has risen to 80% because other issues have been completed quickly without the issue's aggregated time increasing.

$$Time\ progress\ per\ issue\ in\ \% = \frac{(Aggregated\ Time\ Spent)\ h}{(Estimated\ Time\ Issue)\ h}$$

Equation 3 - Time progress per issue in %

Besides the new story point and aggregated time spent column, issue status was included in the calculations. The average time per story point is only calculated with values from resolved issues. The values from the issue status column are used to filter resolved issues.

6.3.3 Implementation

The approach to establish the Delay Detection Mode can be broken down into individual phases. The first step was to copy the Effort Mode of the TimeTrackingTool. This copy was adjusted to suit the naming of the Delay Detection Mode. Then the new static columns were added to display Jira issue information. The next step was to add the dynamic columns that display the calculated estimate result. To fill these dynamic columns the average time per story point was required. Finally, the result of the average time per story point was used to calculate the result in the dynamic column. The overview of these steps is displayed in Figure 13.

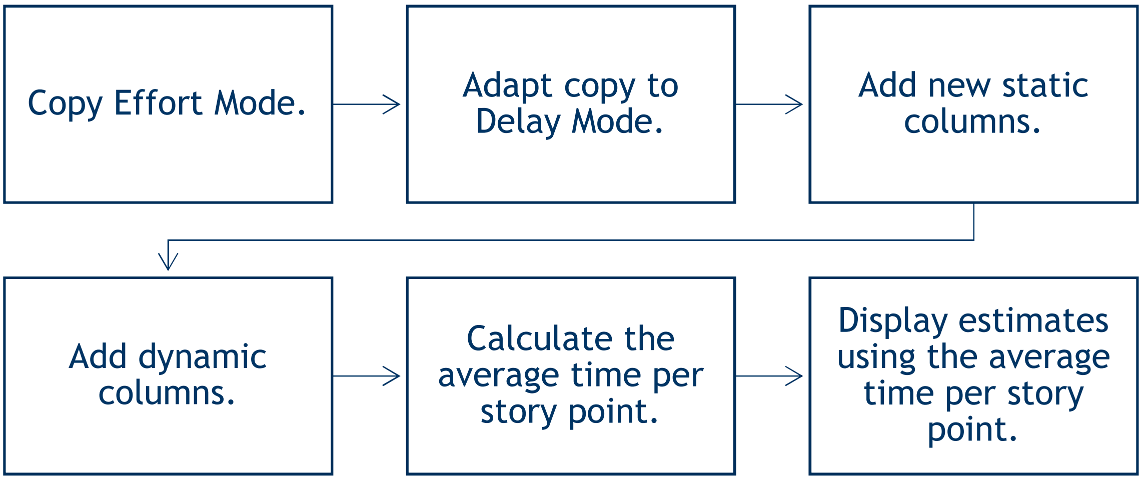

Copy Effort Mode.
Adapt copy to Delay Mode.
Add new static columns.
Add dynamic columns.
Calculate the average time per story point.
Display estimates using the average time per story point.

Figure 13 - Development Approach

To start working on the Delay Detection Mode I needed the TimeTrackingTool project. I had to request permission for the CAS Gerrit server to be able to download the project. After receiving access to the Gerrit server and setting up private and public SSH keys, I was able to git clone the repository.

As intended during the design phase, the Delay Detection Mode will be based on the effort mode of the TimeTrackingTool due to the similarity in used columns and data. I have received an initial briefing of the project from Jakab Fenyovari, one of the main developers responsible for the TimeTrackingTool. The project code does not have inline comments to explain methods or classes. There is also no documentation to provide insight of the functionality either. The lack of comments and documentation has been an obstacle during the initial implementation phase.

IntelliJ has been chosen as the integrated development environment (IDE) because of its out-of-the-box functionalities. IntelliJ’s automation analyses the project execution flow, creates references and builds a syntax tree. IntelliJ has supported me as a developer when examining the TimeTrackingTool project. It assisted by finding implementations and references of classes, methods and variables. For instance, one of the first steps towards developing the Delay Detection Mode was to replicate the LoadEffortModeTask. The replica was changed to the LoadDelayModeTask. The comparison of the LoadEffortModeTask to the LoadDelayModeTask is depicted in Appendix 18. The difference between them was highlighted in blue. Both task files were essential to the respective mode, because of the missing member or missing issue method. The LoadDelayModeTask is testing which issues are not represented in the YAML file. The timetrackingtool.yml contains a list of issues that have been obtained from the Jira server in the last run of the Delay Detection Mode. If the list of available issues from Jira does not equal the list of issues in the timetrackingtool.yml, the timetrackingtool.yml will be synchronised with all available issues from the Jira server. (Cheptsov 2016)

The table below visualises the synchronisation of available issues. Each time running the TimeTrackingTool, the timetrackingtool.yml is synchronised with all Jira issues of the chosen project that contain worklogs.

Available PT Jira issues	Timetrackingtool.yml (before sync)	Timetrackingtool.yml (after sync)
PT-1	-	PT-1
PT-2	PT-2	PT-2
PT-3	PT-3	PT-3
PT-4	-	PT-4

Table 4 - Timetrackingtool.yml synchronised

Copying structural files of the effort mode was the fastest way to create the Delay Detection Mode. All files were then edited to suit the naming and functionality of the Delay Detection Mode.

The greatest challenge and most time-consuming part of the implementation was the calculation and usage of the average hours per story point. For example, the story point, assignee and issue key columns represent static data obtained from Jira. In the effort mode some columns require calculation. The amount and effective hours columns calculate Jira values with static data from the timetrackingtool.yml. The challenge for the average hours per story point was the constant recalculation of the average time for each Jira issue. The final result after the last issue is passed on to every Jira issue data row. This involved working with abstract classes to calculate the average time before displaying the results in the table. Figure 14 explains the current workflow to calculate the average time:

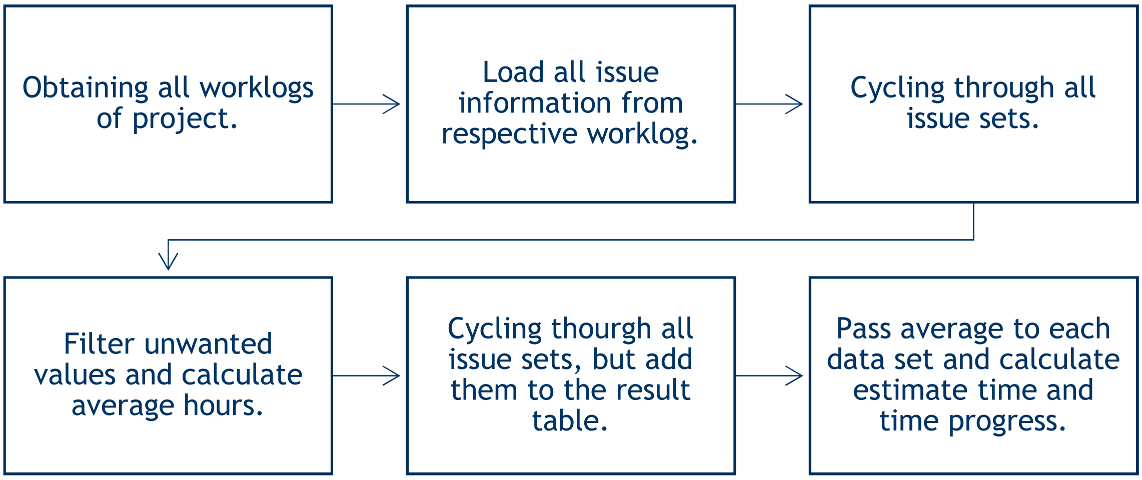

Obtaining all worklogs of project.
Load all issue information from respective worklog.
Cycling through all issue sets.
Filter unwanted values and calculate average hours.
Cycling thourgh all issue sets, but add them to the result table.
Pass average to each data set and calculate estimate time and time progress.

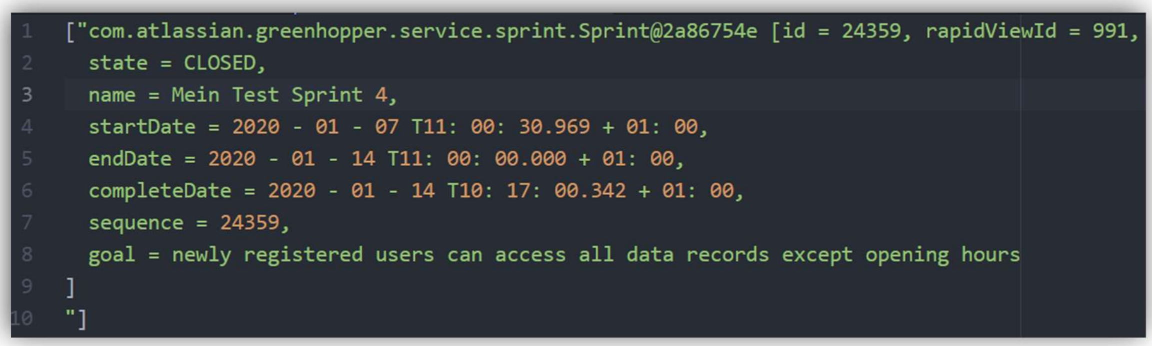
Figure 14 - Abstract workflow of Delay Detection Mode

In the bottom right hand corner of the Delay Detection Mode in Appendix 19, the calculation of the average time per story point is displayed. The sum of story points, worklogs and the average time per story point is displayed to allow the user to understand the calculations made in the table. The display of these numbers has not been part of the wireframe. This feature was designed throughout the implementation process to support the user. The late adaption of this feature is an example of the flexibility offered by using hybrid project management. During the implementation process I decided to implement an additional feature to improve usability. However, the risk of introducing scope creeps, in the form of new features, could delay the project.

6.3.4 Testing

The testing phase took place in two steps. The first step of testing was conducted throughout the implementation. As I am the developer and tester at the same time, I was responsible to guarantee the functionality of each feature in the Delay Detection Mode. For example, after including the sprint column in the result table the displayed data needed to be examined. After running the first test for the sprint column, the sprint column displayed the entire JSON field for sprints. This JSON field contains all information of the sprint.

All sprint information is listed in one data string dividing each field with commas. If a Jira issue was in multiple sprints, each sprint has its own data string. The strings of each sprint will be concatenated to create a longer data set. Regular JSON fields can easily be accessed with get-methods. The only information required of this string is the name of the sprint.



```
["com.atlassian.greenhopper.service.sprint.Sprint@2a86754e [id = 24359, rapidViewId = 991,
  state = CLOSED,
  name = Mein Test Sprint 4,
  startDate = 2020 - 01 - 07 T11: 00: 30.969 + 01: 00,
  endDate = 2020 - 01 - 14 T11: 00: 00.000 + 01: 00,
  completeDate = 2020 - 01 - 14 T10: 17: 00.342 + 01: 00,
  sequence = 24359,
  goal = newly registered users can access all data records except opening hours
]
"]
```

Figure 15 - Sprint JSON field

The approach to this problem was to save each name and sequence to its respective ArrayList. A sprint sequence is unique within a project. The latest sprint name has the highest sequence number. By assigning each sequence number to its name, the latest sprint is accessed by using methods to find the highest sequence number.

Furthermore, unit tests are in place to guarantee the project functionality. The newly implemented data for the Delay Detection Mode was included in existing unit tests. For example, one of the JiraIssueMapTest features is to examine whether the JSON object data is mapped correctly to the Jira issue object. If the data was not mapped correctly a mocked Jira issue data set will be used, and a Jira exception is thrown. These tests have helped to ensure the correct Delay Detection Mode implementation.

The second part of testing occurred after the implementation had been completed. During this phase acceptance and performance tests were conducted. These tests were executed to validate if the project satisfied the specified requirements, user needs and business processes. Acceptance tests were required to replicate real user behaviour to investigate if any errors occur. During these tests I have launched the Delay Detection Mode on three different devices loading different projects. The result was satisfactory because all implemented functionalities passed the functionality test. However, there was an issue with the Delay Detection Mode performance. Unlike the effort mode, the Delay Detection Mode gathers all worklogs of a project. I used a test project with around 20 worklogs during implementation. One of the real projects, with 4,454 worklogs, indicated that the loading time increased significantly. I have tested the software at the CAS office in Karlsruhe, Germany where the Jira servers are located, and also remotely in Perth, Western Australia.

Project worklog size	Location	Loading time in minutes and seconds
20	Karlsruhe	20 sec
	Perth	50 sec
3269	Karlsruhe	1 min 15 sec
	Perth	12 min 25 sec
4454	Karlsruhe	1 min 44 sec
	Perth	15 min 37 sec

Table 5 - Performance test

The table above depicts the significant difference between the locations. The further away the user is from the CAS Jira server, the longer the loading time will be. This effect is significant because of the amount of data that needs to be loaded and calculated by the client. However, the main users of this software will be located in Karlsruhe, Germany, making the location-based performance difference negligible.

Testing included delivering the TimeTrackingTool with the Delay Detection Mode to Marcel Schrumpf as the Product Owner. After installing it on his work computer he was able to use the software without any issues. Marcel has used the software and did not find any problems that require adjustment.

6.3.5 Deployment

After testing was completed successfully, the software project needed to be uploaded to the CAS Gerrit server. Throughout the software development I have submitted multiple commits to maintain different stages of the software project. This helped to return to previous versions in case the project was not functioning after entering new lines of code. These commits were only stored locally, and required to be uploaded to the remote Gerrit server at the end of implementation. The procedure to upload branches and commits to the remote server can have two outcomes.

1. The local branch can be pushed to the Gerrit server repository and remain as an optional feature branch.
2. The local branch is merged into the development branch of the TimeTrackingTool.

If the Delay Detection Mode branch remains as an optional feature branch, the code will not be maintained by other developers. If the Delay Detection Mode feature branch is merged into the development branch, and the TimeTrackingTool code is changing its infrastructure, the Delay Detection Mode code will be adapted with every upgrade to maintain the functionality of the TimeTrackingTool. The difference between both outcomes is the attention of other developers towards the Delay Detection Mode code. As my time at CAS is coming to an end, I will not continue to maintain the Delay Detection Mode code. If the Delay Detection Mode does not function after a major rebase, the other developers will need to invest time to investigate the Delay Detection Mode branch. If the Delay Detection Mode was a part of the TimeTrackingTool development branch, it would always be updated with the TimeTrackingTool. This is because the entire TimeTrackingTool needs to function at all times even after major updates. However, merging the feature branch into the main development branch requires the approval of two developers maintaining the TimeTrackingTool. Acquiring this approval was out of the scope of this thesis. It likely would have cost another ten days of adjusting the Delay Detection Mode to the requirements of the pending approval. The local Delay Detection Mode branch was pushed to the Gerrit server, but remains as an optional feature branch. Every user of the TimeTrackingTool has access to checkout the Delay Detection Mode branch and use the delay detection features.

As a user-friendly the TimeTrackingTool including the Delay Detection Mode is available as a ZIP folder on the CAS WeCloud to all CAS employees. The ZIP folder only requires unpacking to be ready for use. This option does not require the user to have Git version control knowledge.

6.4 Technical Architecture

To explain the abstract technical architecture the Delay Detection Mode is separated into five main components:

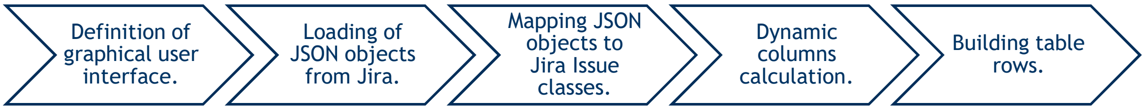
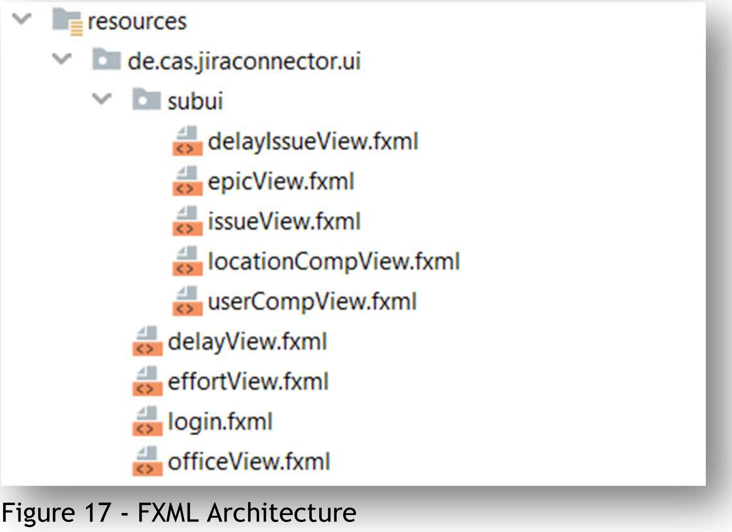

Definition of graphical user interface.
Loading of JSON objects from Jira.
Mapping JSON objects to Jira Issue classes.
Dynamic columns calculation.
Building table rows.

Figure 16 - Architecture of Delay Detection Mode

1. Graphical User Interface

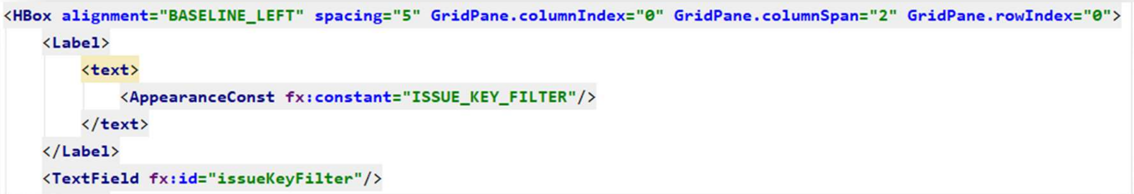
The graphical user interface of the TimeTrackingTool is defined by FXML files. Every mode and tab of the TimeTrackingTool has its own FXML file to define its appearance. The delayView and its sub user interface delayIssueView represent the Delay Detection Mode.



```
resources
  de.cas.jiraconnector.ui
    subui
      delayIssueView.fxml
      epicView.fxml
      issueView.fxml
      locationCompView.fxml
      userCompView.fxml
    delayView.fxml
    effortView.fxml
    login.fxml
    officeView.fxml
```

Figure 17 - FXML Architecture

The delayIssueView is the main component to establish the appearance of the result table. The Delay Detection result table is defined by the delayIssue.fxml. FXML is an XML-based user interface markup language to define the graphical user interface of JavaFX applications. The benefit of using FXML is its hierarchical XML-structure and its support on any Java-supporting device. Figure 18 demonstrates a cut out from the delayIssueView.fxml. The FXML file will contain information to define components like titles, buttons, tabs and appearance of the table.



```
<HBox alignment="BASELINE_LEFT" spacing="5" GridPane.columnIndex="0" GridPane.columnSpan="2" GridPane.rowIndex="0">
    <Label>
        <text>
            <AppearanceConst fx:constant="ISSUE_KEY_FILTER"/>
        </text>
    </Label>
    <TextField fx:id="issueKeyFilter"/>
```

Figure 18 - delayIssueView.fxml

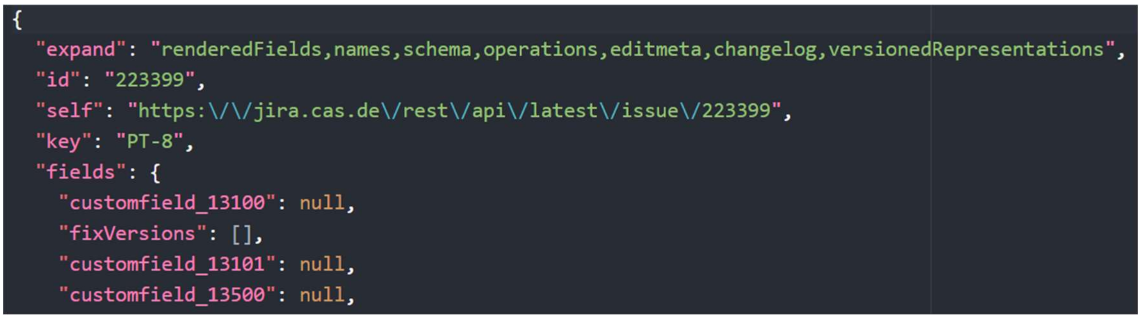
This screenshot is an example of the delayIssueView.fxml content. In this section the issueKeyFilter component is defined. The HBox is similar to a container. It frames the children and allows to define its size and position. The Label is an element to define the title or naming. In this case it is the naming of the TextField issueKeyFilter. The result of the delayIssueView section is in Figure 19.

Figure 19 - Issue Key Filter

2. Loading Jira Server Data

The acquireWorklog method is used to establish the connection to the Jira server and load all JSON objects. Firstly, it examines if all config information is given. Then it tries to establish a connection to the Jira server.

Secondly, the acquireWorklog method of the LoginController class initiates the loading process. This disables all entry fields and displays the loading bars. Once the Jira connection was successful, the next step is to call the UserActionProcessor class. The UserActionProcessor starts the respective TableTask depending on the selected program mode. As the Delay Mode is selected the delayTableTask will be called. The delayTableTask is a Task with the DelayIssueWorklogTable as a variable. Tasks represent a conglomeration of work in JavaFX that can be called and executed. This task is loading all Jira Issue JSON Objects that contain worklogs. These JSON objects are requested from the Jira server with a Http-Get request.



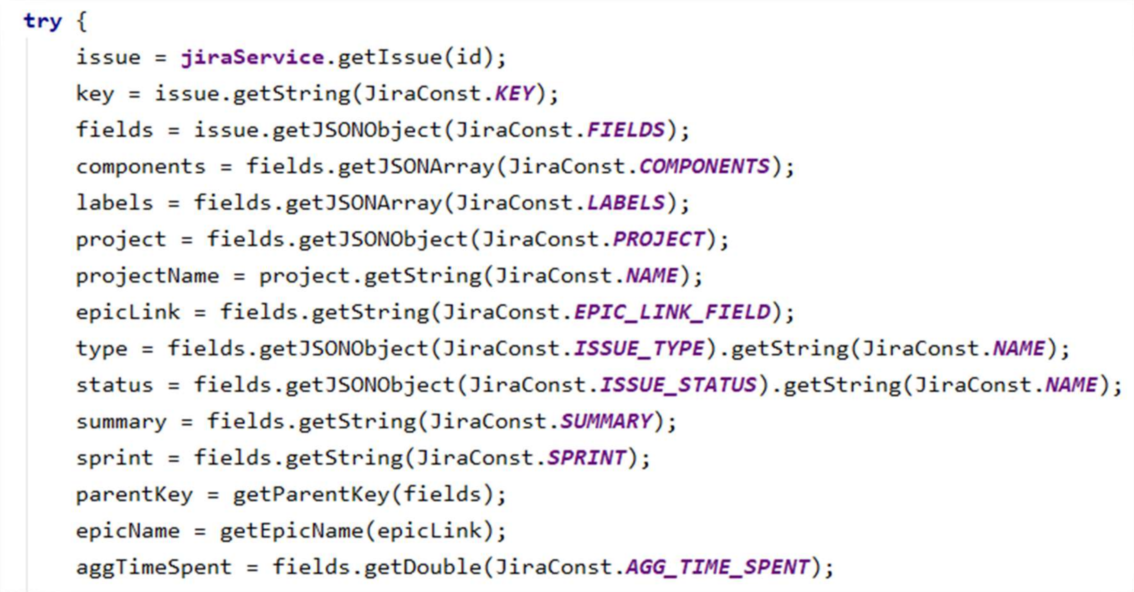
```
{
  "expand": "renderedFields,names,schema,operations,editmeta,changelog,versionedRepresentations",
  "id": "223399",
  "self": "https:\/\/jira.cas.de\/rest\/api\/latest\/issue\/223399",
  "key": "PT-8",
  "fields": {
    "customfield_13100": null,
    "fixVersions": [],
    "customfield_13101": null,
    "customfield_13500": null,
```

Figure 20 - JSON Jira Issue

Figure 20 is a section of a JSON object received from Jira. This object contains all information of a Jira issue. It contains information like the id, issue key and all worklogs on this issue. Every element of a JSON object can be directly accessed to obtain individual parts instead of the entire

3. Mapping JSON Object

To make the the JSON object more usable for the application, the necessary information is mapped by a method in the JiraIssueMap class. In Figure 21 this method is trying to assign parts of the JSON object to a variable. For example, the key variable is obtaining the JSON field key. The field key is always displayed in the JSON object in Figure 20.



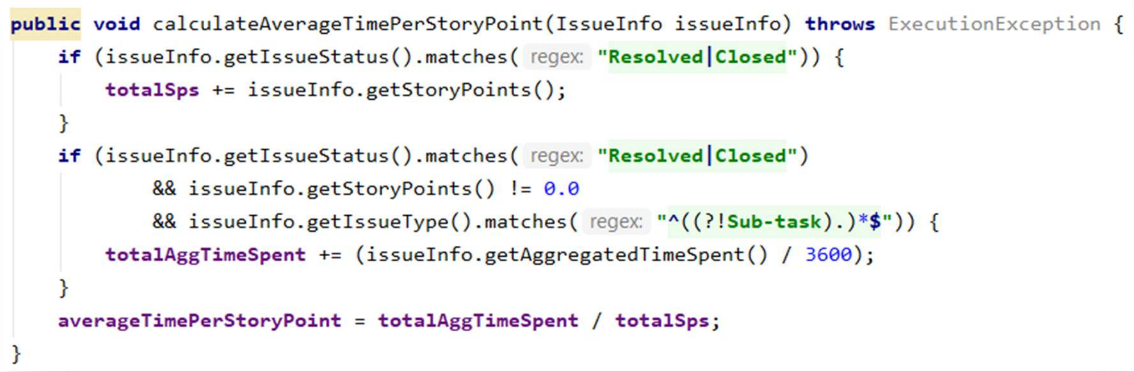
```
try {
    issue = jiraService.getIssue(id);
    key = issue.getString(JiraConst.KEY);
    fields = issue.getJSONObject(JiraConst.FIELDS);
    components = fields.getJSONArray(JiraConst.COMPONENTS);
    labels = fields.getJSONArray(JiraConst.LABELS);
    project = fields.getJSONObject(JiraConst.PROJECT);
    projectName = project.getString(JiraConst.NAME);
    epicLink = fields.getString(JiraConst.EPIC_LINK_FIELD);
    type = fields.getJSONObject(JiraConst.ISSUE_TYPE).getString(JiraConst.NAME);
    status = fields.getJSONObject(JiraConst.ISSUE_STATUS).getString(JiraConst.NAME);
    summary = fields.getString(JiraConst.SUMMARY);
    sprint = fields.getString(JiraConst.SPRINT);
    parentKey = getParentKey(fields);
    epicName = getEpicName(epicLink);
    aggTimeSpent = fields.getDouble(JiraConst.AGG_TIME_SPENT);
```

Figure 21 - Jira Issue Mapping

The mapped variables will be passed on to the IssueInfo list to be accessible for the DelayIssueWorklogTable after the next step.

4. Calculate Estimates

The centrepiece of the Delay Detection Mode was the automated calculation of average times per story points. This calculation is done by iterating through all IssueInfo list elements. The calculateAverageTimePerStoryPoint method is called in a for loop to iterate through all elements of the IssueInfo list. During each iteration the method evaluates if the element is resolved or closed to add the story points to the total story points variable. The next step is to evaluate if the element is resolved, has story points and is not a sub-task. If all three criteria are fulfilled the elements aggregated time spent is counted towards the total. After the last iteration the final division result of total aggregated time spent and total story points equals the average time per story points. The result is passed on to the DelayCalcData class to calculate the estimated time and time progress.

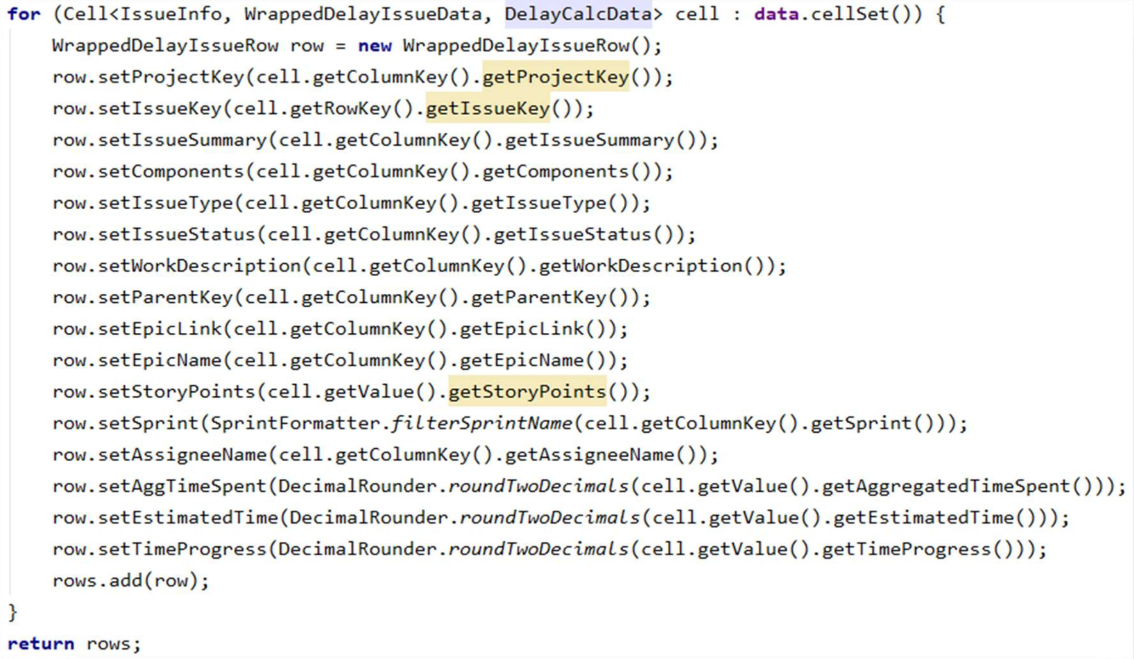


```
public void calculateAverageTimePerStoryPoint(IssueInfo issueInfo) throws ExecutionException {
    if (issueInfo.getIssueStatus().matches( regex: "Resolved|Closed")) {
        totalSps += issueInfo.getStoryPoints();
    }
    if (issueInfo.getIssueStatus().matches( regex: "Resolved|Closed")
            && issueInfo.getStoryPoints() != 0.0
            && issueInfo.getIssueType().matches( regex: "^((?!Sub-task).)*$")) {
        totalAggTimeSpent += (issueInfo.getAggregatedTimeSpent() / 3600);
    }
    averageTimePerStoryPoint = totalAggTimeSpent / totalSps;
}
```

Figure 22 - Calculate Average Time per Story Point

5. Building rows

The DelayIssueWorklogTable is assigning values to each column of a table row. This method is in a for loop to iterate through all elements of the IssueInfo class. The DelayIssueWorklogTable defines which cell variables of the IssueInfo or DelayCalcData. For example, the second last row in Figure 23 obtains the Time Progress from the DelayCalcData.

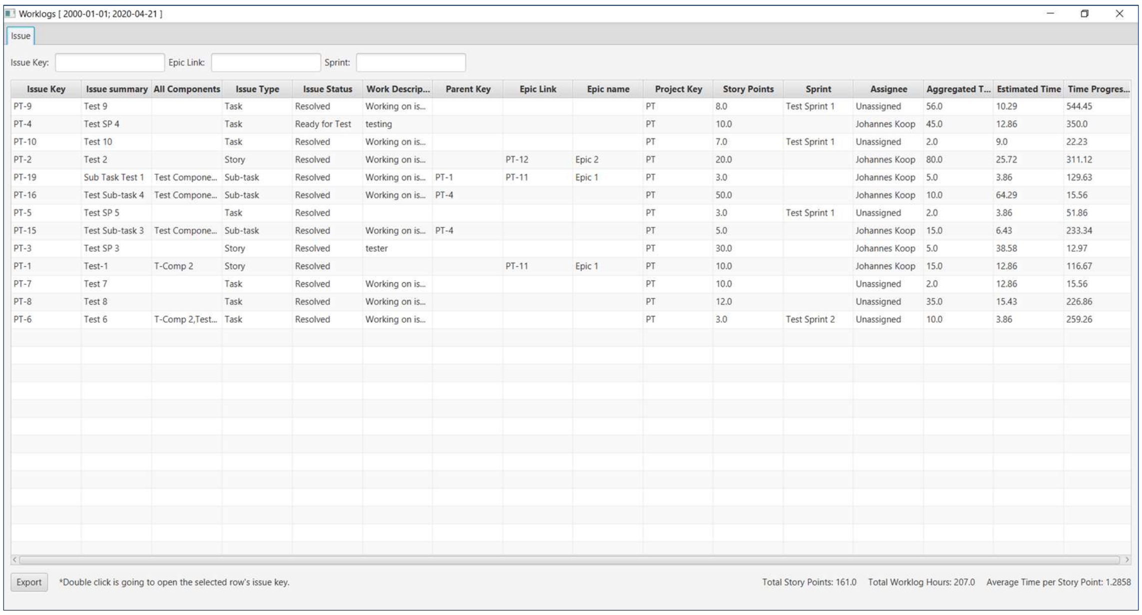


```
for (Cell<IssueInfo, WrappedDelayIssueData, DelayCalcData> cell : data.cellSet()) {
    WrappedDelayIssueRow row = new WrappedDelayIssueRow();
    row.setProjectKey(cell.getColumnKey().getProjectKey());
    row.setIssueKey(cell.getRowKey().getIssueKey());
    row.setIssueSummary(cell.getColumnKey().getIssueSummary());
    row.setComponents(cell.getColumnKey().getComponents());
    row.setIssueType(cell.getColumnKey().getIssueType());
    row.setIssueStatus(cell.getColumnKey().getIssueStatus());
    row.setWorkDescription(cell.getColumnKey().getWorkDescription());
    row.setParentKey(cell.getColumnKey().getParentKey());
    row.setEpicLink(cell.getColumnKey().getEpicLink());
    row.setEpicName(cell.getColumnKey().getEpicName());
    row.setStoryPoints(cell.getValue().getStoryPoints());
    row.setSprint(SprintFormatter.filterSprintName(cell.getColumnKey().getSprint()));
    row.setAssigneeName(cell.getColumnKey().getAssigneeName());
    row.setAggTimeSpent(DecimalRounder.roundTwoDecimals(cell.getValue().getAggregatedTimeSpent()));
    row.setEstimatedTime(DecimalRounder.roundTwoDecimals(cell.getValue().getEstimatedTime()));
    row.setTimeProgress(DecimalRounder.roundTwoDecimals(cell.getValue().getTimeProgress()));
    rows.add(row);
}
return rows;
```

Figure 23 - Assigning Row Values

7. Result

The first sign of the success is the inclusion of the Delay Mode option on the login screen of the TimeTrackingTool. Selecting this option removes the date picker and effort number because neither of them are necessary for this mode. The result table of the Delay Mode Detection is depicted in Figure 19. In the top left corner is are the three filter that allow the user to refine the displayed results by selecting required issue keys, epic links or sprints.


Worklogs [2000-01-01; 2020-04-21]

Issue Key	Issue summary	All Components	Issue Type	Issue Status	Work Descrip...	Parent Key	Epic Link	Epic name	Project Key	Story Points	Sprint	Assignee	Aggregated T...	Estimated Time	Time Progres...
PT-9	Test 9		Task	Resolved	Working on is...				PT	8.0	Test Sprint 1	Unassigned	56.0	10.29	544.45
PT-4	Test SP 4		Task	Ready for Test	testing				PT	10.0		Johannes Koop	45.0	12.86	350.0
PT-10	Test 10		Task	Resolved	Working on is...				PT	7.0	Test Sprint 1	Unassigned	2.0	9.0	22.23
PT-2	Test 2		Story	Resolved	Working on is...		PT-12	Epic 2	PT	20.0		Johannes Koop	80.0	25.72	311.12
PT-19	Sub Task Test 1	Test Compone...	Sub-task	Resolved	Working on is...	PT-1	PT-11	Epic 1	PT	3.0		Johannes Koop	5.0	3.86	129.63
PT-16	Test Sub-task 4	Test Compone...	Sub-task	Resolved	Working on is...	PT-4			PT	50.0		Johannes Koop	10.0	64.29	15.56
PT-5	Test SP 5		Task	Resolved					PT	3.0	Test Sprint 1	Unassigned	2.0	3.86	51.86
PT-15	Test Sub-task 3	Test Compone...	Sub-task	Resolved	Working on is...	PT-4			PT	5.0		Johannes Koop	15.0	6.43	233.34
PT-3	Test SP 3		Story	Resolved	tester				PT	30.0		Johannes Koop	5.0	38.58	12.97
PT-1	Test-1	T-Comp 2	Story	Resolved			PT-11	Epic 1	PT	10.0		Johannes Koop	15.0	12.86	116.67
PT-7	Test 7		Task	Resolved	Working on is...				PT	10.0		Unassigned	2.0	12.86	15.56
PT-8	Test 8		Task	Resolved	Working on is...				PT	12.0		Unassigned	35.0	15.43	226.86
PT-6	Test 6	T-Comp 2,Test...	Task	Resolved	Working on is...				PT	3.0	Test Sprint 2	Unassigned	10.0	3.86	259.26

*Double click is going to open the selected row's issue key.

Total Story Points: 161.0 Total Worklog Hours: 207.0 Average Time per Story Point: 1.2858

Figure 24 - Result Table Delay Detection Mode

The columns are displaying their static values to display relevant issue information. The aggregated time spent, estimated time and time progress column are displaying dynamic results. Every row represents an individual Jira issue. The user can use the values to interpret the development of individual Jira issues.

In the bottom left the export button is located which allows the user to export all values in a CSV file. This allows the usage of the results in further calculations. In the bottom right of the table the two variables for the average time per story point is located. The sums of all resolved issue's worklogs and story points are displayed and the therefore resulting average time per story point. These variables allow the user to understand the calculation in the dynamic columns. Furthermore, the columns can be rearranged by drag and drop. This can be useful to change the order for CSV exports or to have a specific order of columns.

Ultimately, the tool met all client requirements. The tool reduced the delay investigation process from 40 minutes to approximately 5 minutes. This is a time saving of 87,5%. This time saving will allow the project manager to use the tool more frequently. The Delay Detection Mode will help Project Managers to receive instantaneous issue progress results. The Aggregated Time Spent will allow to understand how much time was spent in total. The Estimated

Time is the prediction of the total amount of hours spent. The Time Progress is the quotient between Aggregated time Spent and Estimated Time to interpret if the actual time has exceeded the predicted time.

8. Future improvement

Software develops over time, and so will the Delay Detection Mode. I have had ideas for future improvements that will enhance the performance of the Delay Detection Mode, or add new functionalities. All of these improvements were out of scope for the thesis project.

8.1 Saving time progress and average time spent results

Each time the user is running the Delay Detection Mode the time progress result is calculating with the latest update of aggregated time spent and the latest average time per story point. All values used in the time progress calculation will change over the life time of the project. To give each result more meaning, the user should be able to see the development of the variables over time. Each time running the Delay Detection Mode of the TimeTrackingTool the value of the average time per story points could be stored in an Excel spreadsheet that visualises the development of the average time per story point in a line graph chart. The time progress values could be listed in a table that has time stamps for each time progress value of each Jira issue.

Keeping this information in store adds value to the time progress column. The Project Manager can compare the development of a Jira issue and make a more informed decision on allocating resources, because the user sees what trend the Jira issue is following. The issue might have been increasing or decreasing its performance since running the application last time. For example, the Project Manager can interpret the trend of the Time Progress when previous results are considered. Table 8 demonstrates the trend of Issue-1 and Issue-2. Issue-1 Time Progress has increased steadily over the last three months, which indicates normal development. However, the Issue-2 Time Progress had a significant spike throughout March which lead to exceeding 100%. The significant spike of Issue-2 could be an indicator of development difficulties which need further investigation. Without the previous result this spike would not be visible.

Time Stamp	01.02.2020	01.03.2020	01.04.2020
Jira issue-1	80%	90%	95%
Jira issue-2	75%	70%	110%

Table 6 - Saving average time results

Alternatively, the previous results of the Time Progress could be displayed in a separate tab. The results tab would contain the same information as the Excel file, but contain it inside of the application. Additionally, the data of this tab could be displayed in a graph chart in a separate tab. Figure 20 depicts the potential look of the graphical time progress representation. This would give the Project Manager a visual analysis of the issue development. The Project Manager can see if the Time Progress is increasing rapidly in comparison to the other graphs which would indicate a risk of delays.

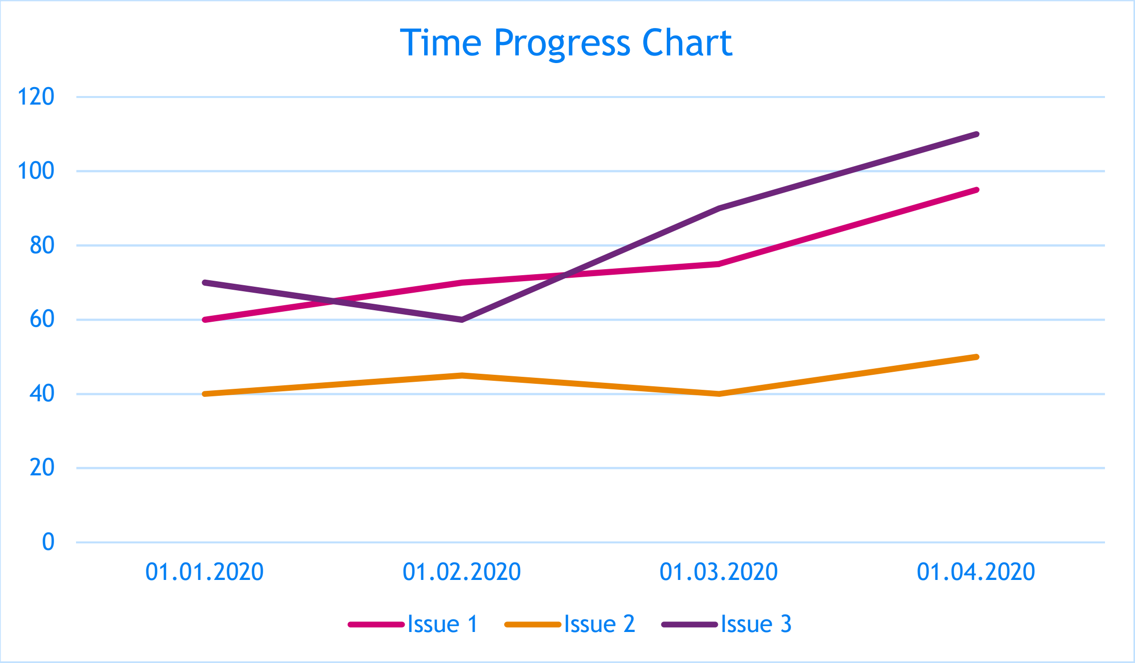


Figure 25 - Time Progress Chart Concept

8.2 Colour coding the time progress bar

The time progress value gives the user information about the relationship between time spent and time planned. If the value exceeds 100, it indicates that the time spent outgrew the time planned. The cells could be colour coded to improve the user experience when searching for critical issues that are close to exceeding, or have exceeded, their time planned.

Time Progress in %	0-85%	85-100%	>100%

Table 7 - Time progress colour coded

Colour coded cells would stand out and improve the visibility of critical Jira issues. For example, all values below 85% would be green, indicating that the time spent has not exceeded the planned time. If the issue status is not resolved and time progress is between 85% - 100% the cells could be yellow. This would highlight Jira issues that need to be watched as they are getting closer to exceeding the time planned. If the issues status is not resolved, and time progress is higher than 100%, the cells could be red to indicate the need of root cause investigations.

8.3 Removal of worklogs

The effort mode of the TimeTrackingTool obtains all worklogs from Jira for the given time period and project key. For all worklogs the effort mode loads the respective Jira issue data. At this point the Delay Detection Mode is still obtaining all worklogs. However, it does not display any worklog related data. It only contains issue relevant information. To improve the performance of the Delay Detection Mode and remove unnecessary code, the Delay Detection Mode should obtain all Jira issues from a project instead of the worklogs. This would increase the performance of the Delay Detection Mode by removing the loading of worklogs. Loading all worklogs first, and not displaying their information, is unnecessary and wastes resources.

8.4 Sprint listing multiple items

The sprint column provides the user with more information about the Jira issue. The sprint filter uses the respective column to filter all issues by entering a sprint of choice. Currently only the latest sprint will be displayed. In Appendix 19 the “All Components” column contains multiple entries per cell. At the same time the components filter allows the user to enter a component. All rows that contain the component will be displayed, though it may have other components in the same cell. This feature could be useful for the sprint column as the current sprint, and all previous sprints, are displayed. This would allow the user to find specific information about a past sprint. However, selecting a past sprint would still display the latest amount of aggregated time spent.

8.5 Concatenating percentage to time progress

To improve readability of the time progress values in the Delay Detection Mode a percentage symbol could be concatenated to each value. Currently only the column title “Time Progress in %” contains the information that displayed values are to be interpreted in percentages. I have tried to add the percentage symbol during the implementation of the Delay Detection Mode, but did not find a simple approach because the time progress value is a Java type double. Converting this value to a string before displaying it and concatenating the percentage symbol could be an approach. Adding the percentage symbol would decrease the likelihood of the value being misunderstood.

Time Progress in % (Current)	Time Progress in % (Improved)

55,87	55,87%

Table 8 - Time progress improvement

8.6 Merge Delay Detection Mode to development branch

As discussed in 5.2.5, the local feature branch created for the Delay Detection Mode was pushed to the Gerrit server, but remains an independent feature branch. This means that the Delay Detection Mode feature branch does not benefit from any updates for the main TimeTrackingTool branch. Figure 21 is depicting a simplified version of the current Git structure. The Delay Detection Mode feature branch was created from the main branch but was never merged back to the main branch. As a result, the Delay Detection Mode branch does not obtain the updates on the main branch.

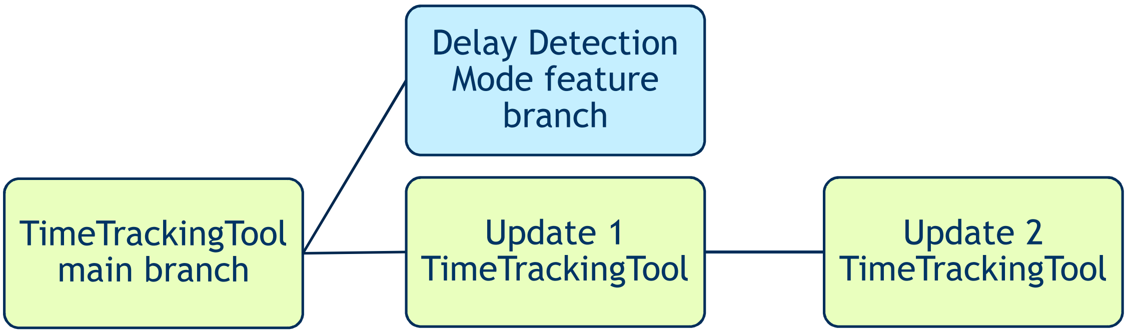

Delay Detection Mode feature branch
TimeTrackingTool main branch
Update 1 TimeTrackingTool
Update 2 TimeTrackingTool

Figure 26 - Git Branch

A goal for this feature is for it to be included in the standard TimeTrackingTool version. Merging the feature branch to the development branch will remove the need for developers to rebase the feature branch when the development branch is updated. This will guarantee that Delay Detection Mode users can use the latest features of the TimeTrackingTool, without rebasing and merging the feature branch themselves. For example, the developers may have updated the TimeTrackingTool to use the Jira REST API to load all data. If the Delay Detection Mode was merged to the development branch, the developers will provide the changes to the entire project including the Delay Detection Mode. So, the user will only need to git pull the latest changes of the development branch.

Having said that, if the Delay Detection Mode is on the feature branch, the user will need to rebase the feature branch to access the new REST API features. A major rebase will most likely cause merge conflicts that require manual solving. If the user is not familiar with the code and Git, a developer will need to help the user to resolve the merge conflicts. As a consequence, merging the Delay Detection Mode branch to the development branch will prevent unnecessary support expenses.

9. Conclusion

Changing business processes can be a difficult task. Stakeholders need to be convinced that new processes are beneficial, and feasible. Researching the teamwork characteristics of CAS developers has helped to find an incentive for this project. Interviewing Steffen Euch and Marcel Schrumpf has supported the creation of the Delay Detection Mode wireframe, and subsequent goals. Stakeholders of the Delay Detection Mode project have been convinced by its usefulness and simplicity. The Delay Detection Mode avoids carelessness errors and saves time by removing manual steps for the Project Manager. The thesis project has been successfully completed within the given time frame. Marcel Schrumpf, as the main client, has received and accepted the results. He is enthusiastic about the project management support tool as it met all of his requirements for this thesis project. The use of this tool is allowing him to detect delays quickly and more frequently.

As an intern and working student, I was able to support Marcel Schrumpf in his project management tasks. I enjoyed calculating and interpreting results for project controlling reports. I am glad that I was able to optimise CAS project management processes with my thesis.

After working for CAS as an intern, working student and now writing my bachelor’s thesis, I was able to develop new skills and establish business and personal connections. I want to personally thank Marcel Schrumpf for his continuous support throughout all stages of my time at CAS.

References

Electronic sources

2008. A Guide To The Project Management Body Of Knowledge (PMBOK® Guide). 4th ed. [ebook] Newtown Square: A Guide to the Project Management Body of Knowledge (PMBOK® Guide). Available at: <https://ebookcentral.proquest.com/> [Accessed 19 May 2020].

Ardem. 2017. Why 'Good Enough' Isn't Enough When It Comes To Data Entry. [online] Available at: <https://ardem.com/data-entry-services/why-good-enough-isnt-enough-for-data-entry/> [Accessed 3 April 2020].

Atlassian. (n.d.). Development and Collaboration Software Company | Atlassian. [online] Available at: https://www.atlassian.com/company [Accessed 15 Jan. 2020].

Bhat, A., n.d. Leading Questions: Definition And Characteristics With Examples. [online] QuestionPro. Available at: <https://www.questionpro.com/blog/leading-questions/> [Accessed 14 April 2020].

Cas.de. (2014). Zukunft souverän gestalten mit "Software Made in Germany" | CAS Software AG. [online] Available at: https://www.cas.de/nc/de/kundenzeitschrift/details/article/zukunft-souveraen-gestalten-mit-software-made-in-germany.html [Accessed 15 Jan. 2020].

Cas.de. (n.d.). History | CAS Software AG. [online] Available at: https://www.cas.de/en/company/cas-software-ag/profile-history/history.html [Accessed 15 Jan. 2020].

Cheptsov, A., 2016. Enjoying Java And Being More Productive With Intellij IDEA | Intellij IDEA Blog. [online] IntelliJ IDEA Blog. Available at: <https://blog.jetbrains.com/idea/2016/03/enjoying-java-and-being-more-productive-with-intellij-idea/> [Accessed 10 April 2020].

Cruz, F. (2019). Herman Hollerith Tabulating Machine. [online] Columbia.edu. Available at: http://www.columbia.edu/cu/computinghistory/hollerith.html [Accessed 15 Jan. 2020].

Datanyze.com. (n.d.). Jira Market Share and Competitor Report | Compare to Jira, Microsoft Project, Smartsheet. [online] Available at: https://www.datanyze.com/market-share/project-management/jira-market-share [Accessed 15 Jan. 2020].

Jira.cas.de. n.d. CAS - Jira. [online] Available at: <https://jira.cas.de> [Accessed 3 April 2020].

Kramer, S., Mueller, J., Amabile, T., Simpson, W., Hadley, C. and Fleming, L., 2002. Time Pressure And Creativity In Organizations: A Longitudinal Field Study. Ph. D. Harvard Business School, Harvard University.

LaMarca, N., 2011. The Likert Scale: Advantages And Disadvantages. [online] Field Research in Organizational Psychology. Available at: <https://psyc450.wordpress.com/2011/12/05/the-likert-scale-advantages-and-disadvantages/> [Accessed 15 April 2020].

Mockflow.com. 2020. Mockflow - Wireframe Tools, Prototyping Tools, UI Mockups, UX Suite, Remote Designing. [online] Available at: <https://www.mockflow.com/> [Accessed 8 April 2020].

Pawlan, M., 2013. What Is Javafx? | Javafx 2 Tutorials And Documentation. [online] Docs.oracle.com. Available at: <https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm> [Accessed 20 April 2020].

Pittet, S., n.d. The Different Types Of Testing In Software | Atlassian. [online] Atlassian. Available at: <https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing> [Accessed 13 April 2020].

Radigan, D., n.d. What Are Story Points And How Do You Estimate Them?. [online] Atlassian. Available at: <https://www.atlassian.com/agile/project-management/estimation> [Accessed 9 April 2020].

Robins, D., 2017. Hybrid: A New Project Management Approach. [online] CIO. Available at: <https://www.cio.com/article/3222872/hybrid-a-new-project-management-approach.html> [Accessed 8 April 2020].

Rongala, A., 2015. Manual Data Entry Challenges. [online] Invensis Technologies. Available at: <https://www.invensis.net/blog/data-processing/top-6-manual-data-entry-challenges-companies-face/> [Accessed 3 April 2020].

West, D., n.d. Agile Scrum Roles | Atlassian. [online] Atlassian. Available at: <https://www.atlassian.com/agile/scrum/roles> [Accessed 8 April 2020].

Westland, J., 2018. The Triple Constraint In Project Management: Time, Scope & Cost. [online] ProjectManager.com. Available at: <https://www.projectmanager.com/blog/triple-constraint-project-management-time-scope-cost> [Accessed 20 April 2020].

Interviews

Euch, S., 2019. Usage Of The Timetrackingtool.

Schrumpf, M., 2019. Requirement Analysis For Timetrackingtool.

Figures

Tables

Equations

Appendices

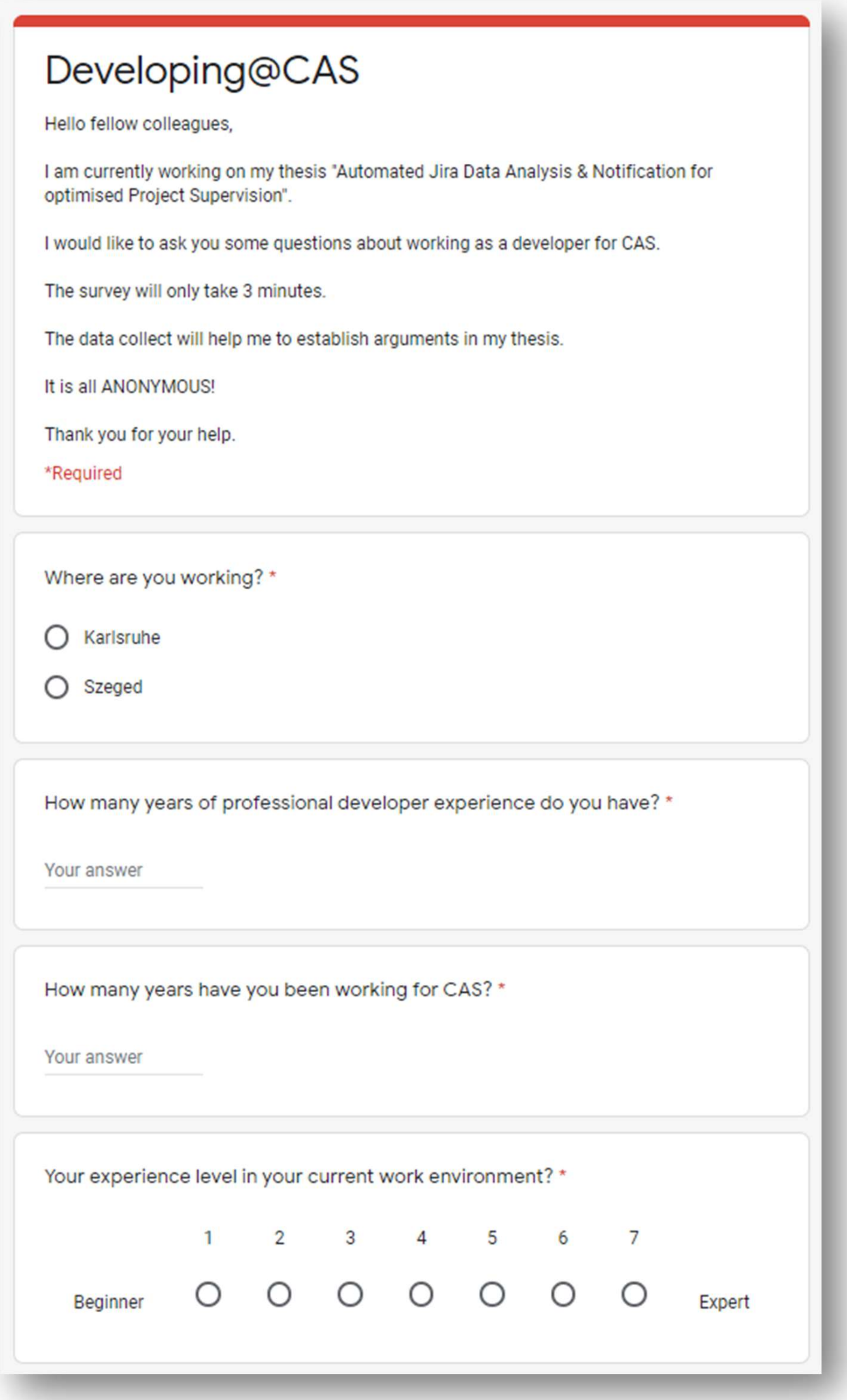
Confidentiality clause:

Appendices 1-10 contain confidential data of CAS Software AG.

These appendices may only be made available to the first and second reviewers and authorized members of the board of examiners. Any publication and duplication of the confidential appendices - even in part - is prohibited.

An inspection of the confidential appendices by third parties requires the expressed permission of the author and CAS Software AG.

Appendix 11: Survey Questionnaire



Developing@CAS

Hello fellow colleagues,

I am currently working on my thesis "Automated Jira Data Analysis & Notification for optimised Project Supervision".

I would like to ask you some questions about working as a developer for CAS.

The survey will only take 3 minutes.

The data collect will help me to establish arguments in my thesis.

It is all ANONYMOUS!

Thank you for your help.

*Required

Where are you working? *

○ Karlsruhe

○ Szeged

How many years of professional developer experience do you have? *

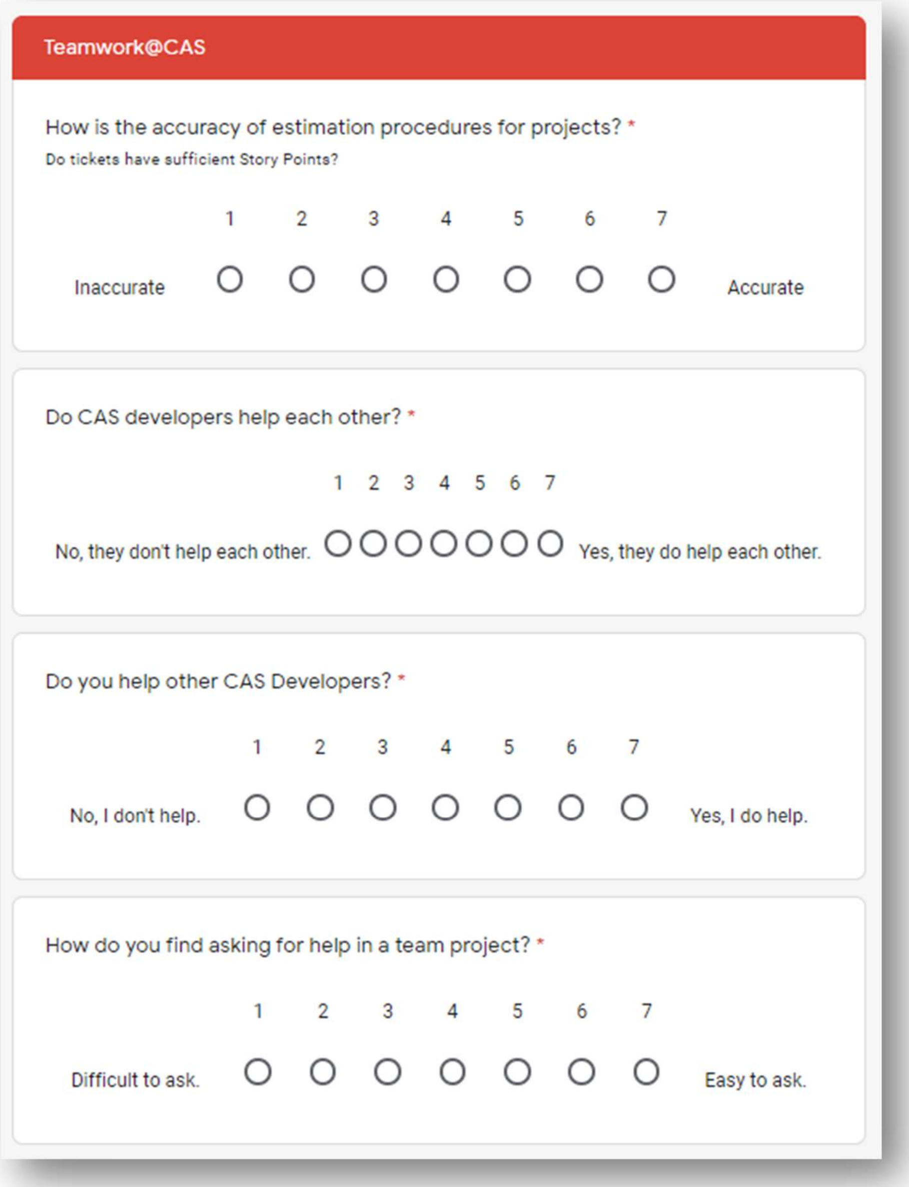
Your answer

How many years have you been working for CAS? *

Your answer

Your experience level in your current work environment? *

	1	2	3	4	5	6	7	
Beginner	○	○	○	○	○	○	○	Expert



Teamwork@CAS

How is the accuracy of estimation procedures for projects? *

Do tickets have sufficient Story Points?

	1	2	3	4	5	6	7	
Inaccurate	○	○	○	○	○	○	○	Accurate

Do CAS developers help each other? *

	1	2	3	4	5	6	7	
No, they don't help each other.	○	○	○	○	○	○	○	Yes, they do help each other.

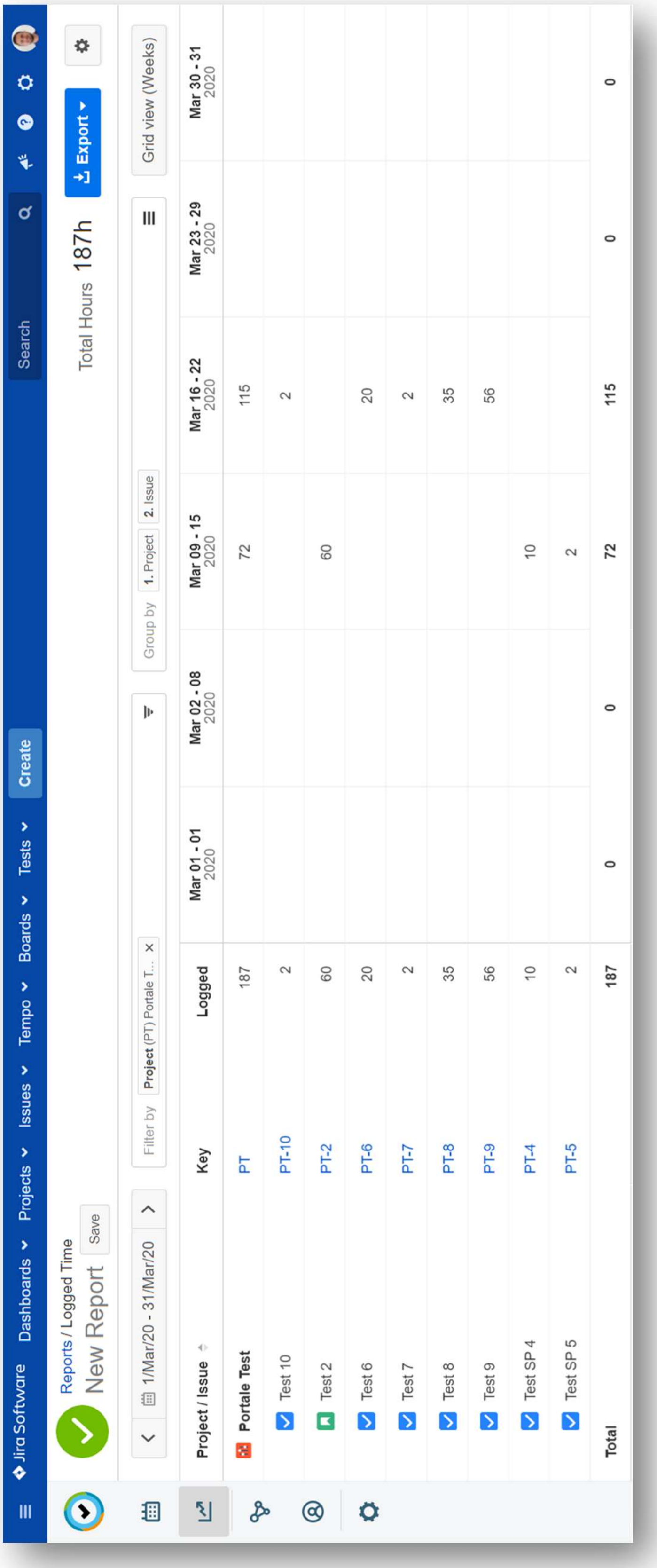
Do you help other CAS Developers? *

	1	2	3	4	5	6	7	
No, I don't help.	○	○	○	○	○	○	○	Yes, I do help.

How do you find asking for help in a team project? *

	1	2	3	4	5	6	7	
Difficult to ask.	○	○	○	○	○	○	○	Easy to ask.

Appendix 12: Jira Tempo Logged Time Report



New Report

Total Hours 187h

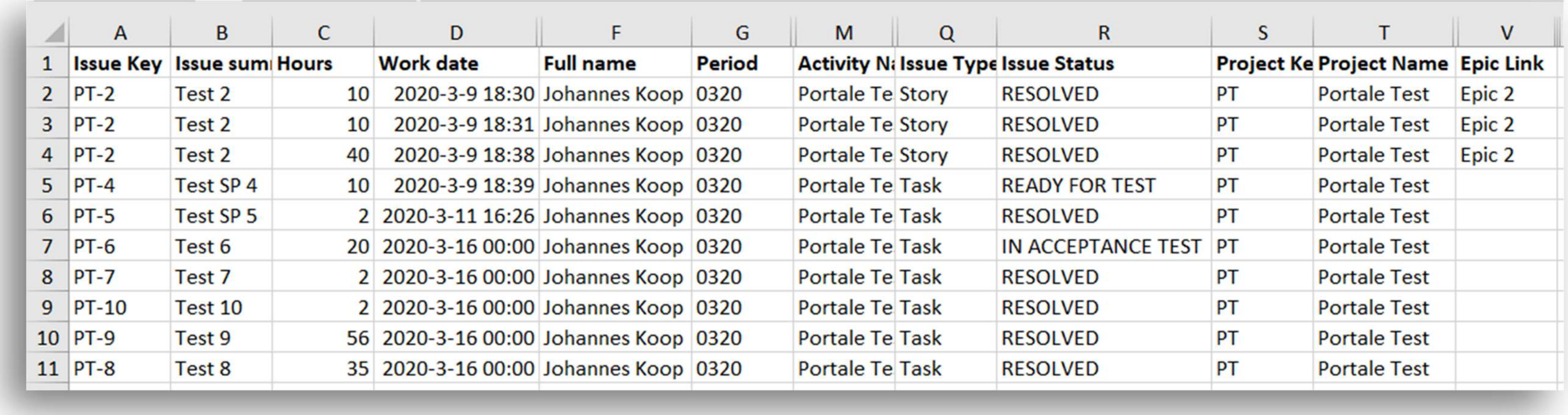
1/Mar/20 - 31/Mar/20

Filter by Project (PT) Portale T... ×

Group by 1. Project 2. Issue

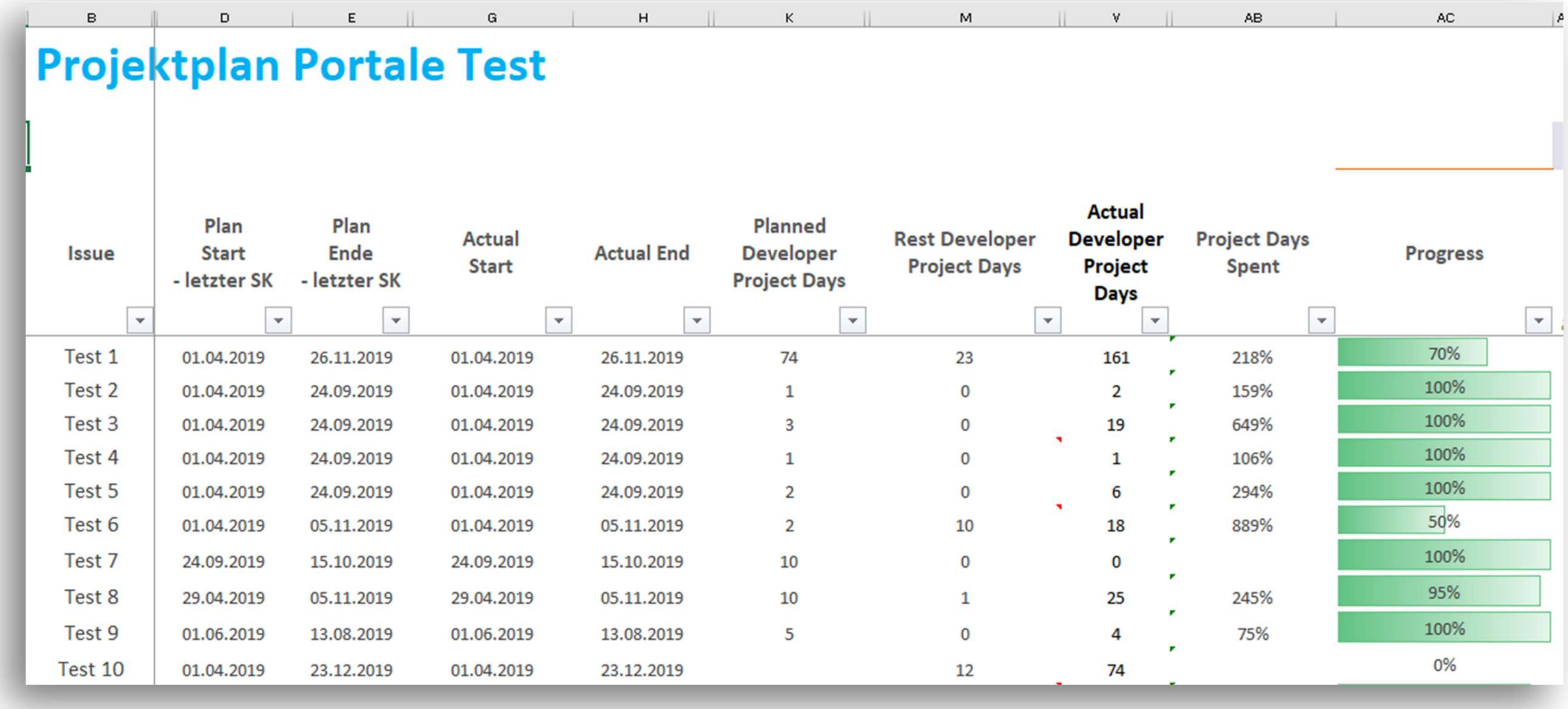
Project / Issue	Key	Logged	Mar 01 - 01 2020	Mar 02 - 08 2020	Mar 09 - 15 2020	Mar 16 - 22 2020	Mar 23 - 29 2020	Mar 30 - 31 2020
Portale Test	PT	187			72	115		
Test 10	PT-10	2				2		
Test 2	PT-2	60			60			
Test 6	PT-6	20				20		
Test 7	PT-7	2				2		
Test 8	PT-8	35				35		
Test 9	PT-9	56				56		
Test SP 4	PT-4	10			10			
Test SP 5	PT-5	2			2			
Total		187	0	0	72	115	0	0

Appendix 13: Jira Tempo Excel Export



	A	B	C	D	F	G	M	Q	R	S	T	V
1	**Issue Key**	**Issue sum**	**Hours**	**Work date**	**Full name**	**Period**	**Activity N**	**Issue Type**	**Issue Status**	**Project Ke**	**Project Name**	**Epic Link**
2	PT-2	Test 2	10	2020-3-9 18:30	Johannes Koop	0320	Portale Te	Story	RESOLVED	PT	Portale Test	Epic 2
3	PT-2	Test 2	10	2020-3-9 18:31	Johannes Koop	0320	Portale Te	Story	RESOLVED	PT	Portale Test	Epic 2
4	PT-2	Test 2	40	2020-3-9 18:38	Johannes Koop	0320	Portale Te	Story	RESOLVED	PT	Portale Test	Epic 2
5	PT-4	Test SP 4	10	2020-3-9 18:39	Johannes Koop	0320	Portale Te	Task	READY FOR TEST	PT	Portale Test	
6	PT-5	Test SP 5	2	2020-3-11 16:26	Johannes Koop	0320	Portale Te	Task	RESOLVED	PT	Portale Test	
7	PT-6	Test 6	20	2020-3-16 00:00	Johannes Koop	0320	Portale Te	Task	IN ACCEPTANCE TEST	PT	Portale Test	
8	PT-7	Test 7	2	2020-3-16 00:00	Johannes Koop	0320	Portale Te	Task	RESOLVED	PT	Portale Test	
9	PT-10	Test 10	2	2020-3-16 00:00	Johannes Koop	0320	Portale Te	Task	RESOLVED	PT	Portale Test	
10	PT-9	Test 9	56	2020-3-16 00:00	Johannes Koop	0320	Portale Te	Task	RESOLVED	PT	Portale Test	
11	PT-8	Test 8	35	2020-3-16 00:00	Johannes Koop	0320	Portale Te	Task	RESOLVED	PT	Portale Test	

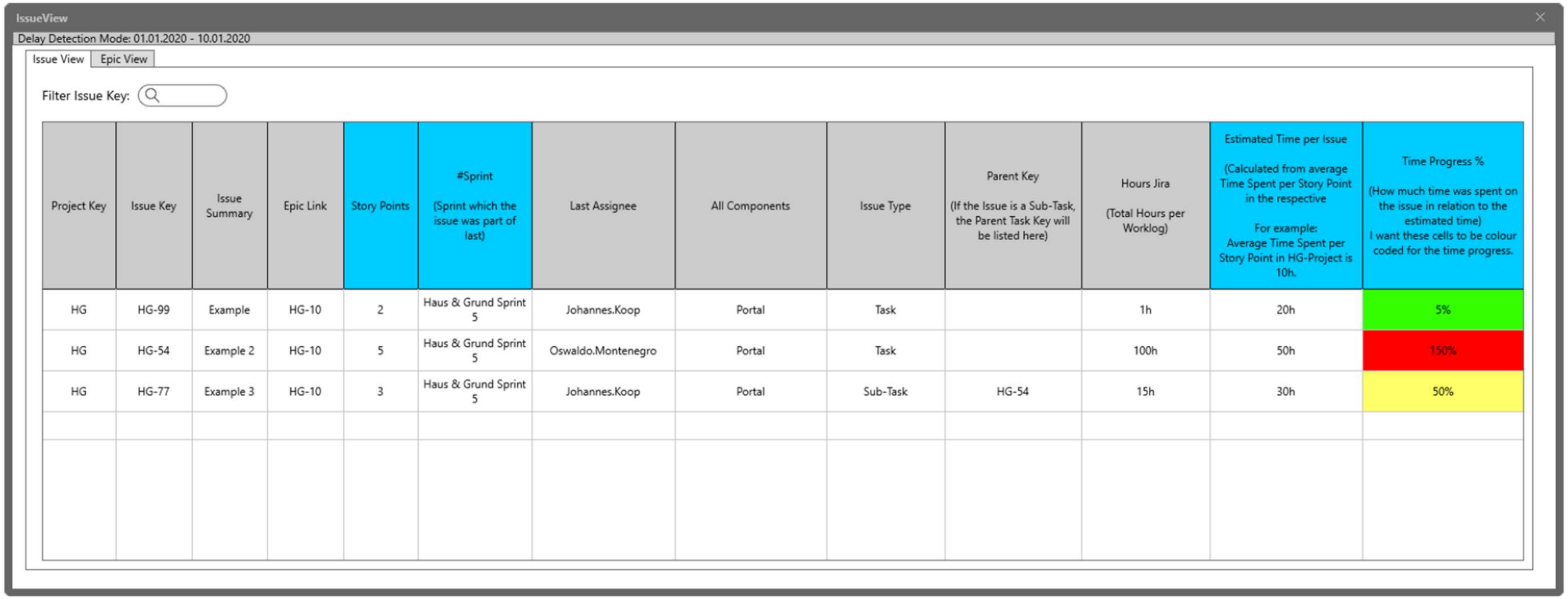
Appendix 14: Project Plan



Projektplan Portale Test

Issue	Plan Start - letzter SK	Plan Ende - letzter SK	Actual Start	Actual End	Planned Developer Project Days	Rest Developer Project Days	Actual Developer Project Days	Project Days Spent	Progress
Test 1	01.04.2019	26.11.2019	01.04.2019	26.11.2019	74	23	161	218%	70%
Test 2	01.04.2019	24.09.2019	01.04.2019	24.09.2019	1	0	2	159%	100%
Test 3	01.04.2019	24.09.2019	01.04.2019	24.09.2019	3	0	19	649%	100%
Test 4	01.04.2019	24.09.2019	01.04.2019	24.09.2019	1	0	1	106%	100%
Test 5	01.04.2019	24.09.2019	01.04.2019	24.09.2019	2	0	6	294%	100%
Test 6	01.04.2019	05.11.2019	01.04.2019	05.11.2019	2	10	18	889%	50%
Test 7	24.09.2019	15.10.2019	24.09.2019	15.10.2019	10	0	0		100%
Test 8	29.04.2019	05.11.2019	29.04.2019	05.11.2019	10	1	25	245%	95%
Test 9	01.06.2019	13.08.2019	01.06.2019	13.08.2019	5	0	4	75%	100%
Test 10	01.04.2019	23.12.2019	01.04.2019	23.12.2019		12	74		0%

Appendix 15: Wireframe

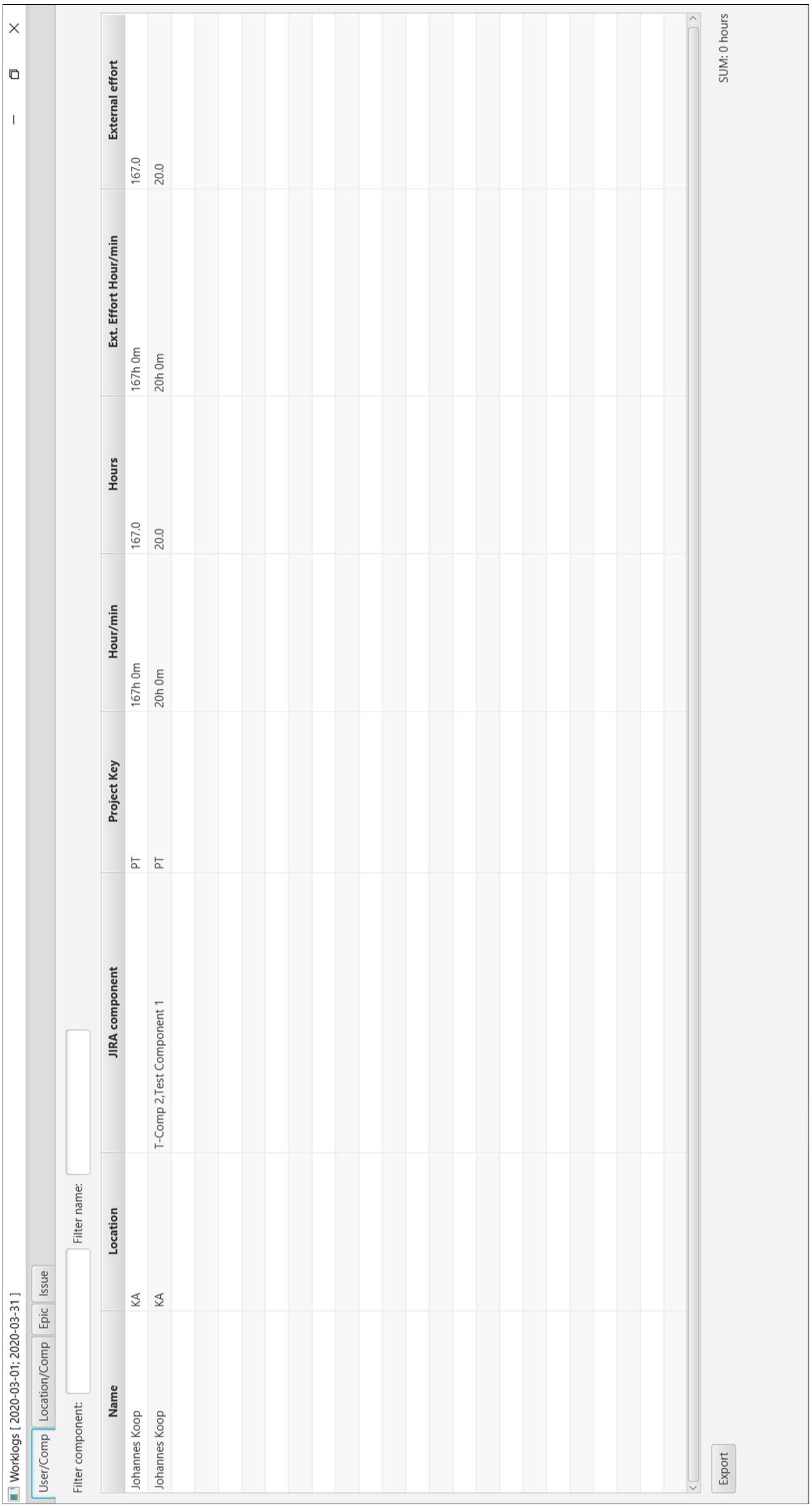


Delay Detection Mode: 01.01.2020 - 10.01.2020

Filter Issue Key:

Project Key	Issue Key	Issue Summary	Epic Link	Story Points	#Sprint (Sprint which the issue was part of last)	Last Assignee	All Components	Issue Type	Parent Key (If the Issue is a Sub-Task, the Parent Task Key will be listed here)	Hours Jira (Total Hours per Worklog)	Estimated Time per Issue (Calculated from average Time Spent per Story Point in the respective For example: Average Time Spent per Story Point in HG-Project is 10h.	Time Progress % (How much time was spent on the issue in relation to the estimated time) I want these cells to be colour coded for the time progress.
HG	HG-99	Example	HG-10	2	Haus & Grund Sprint 5	Johannes.Koop	Portal	Task		1h	20h	5%
HG	HG-54	Example 2	HG-10	5	Haus & Grund Sprint 5	Oswaldo.Montenegro	Portal	Task		100h	50h	150%
HG	HG-77	Example 3	HG-10	3	Haus & Grund Sprint 5	Johannes.Koop	Portal	Sub-Task	HG-54	15h	30h	50%

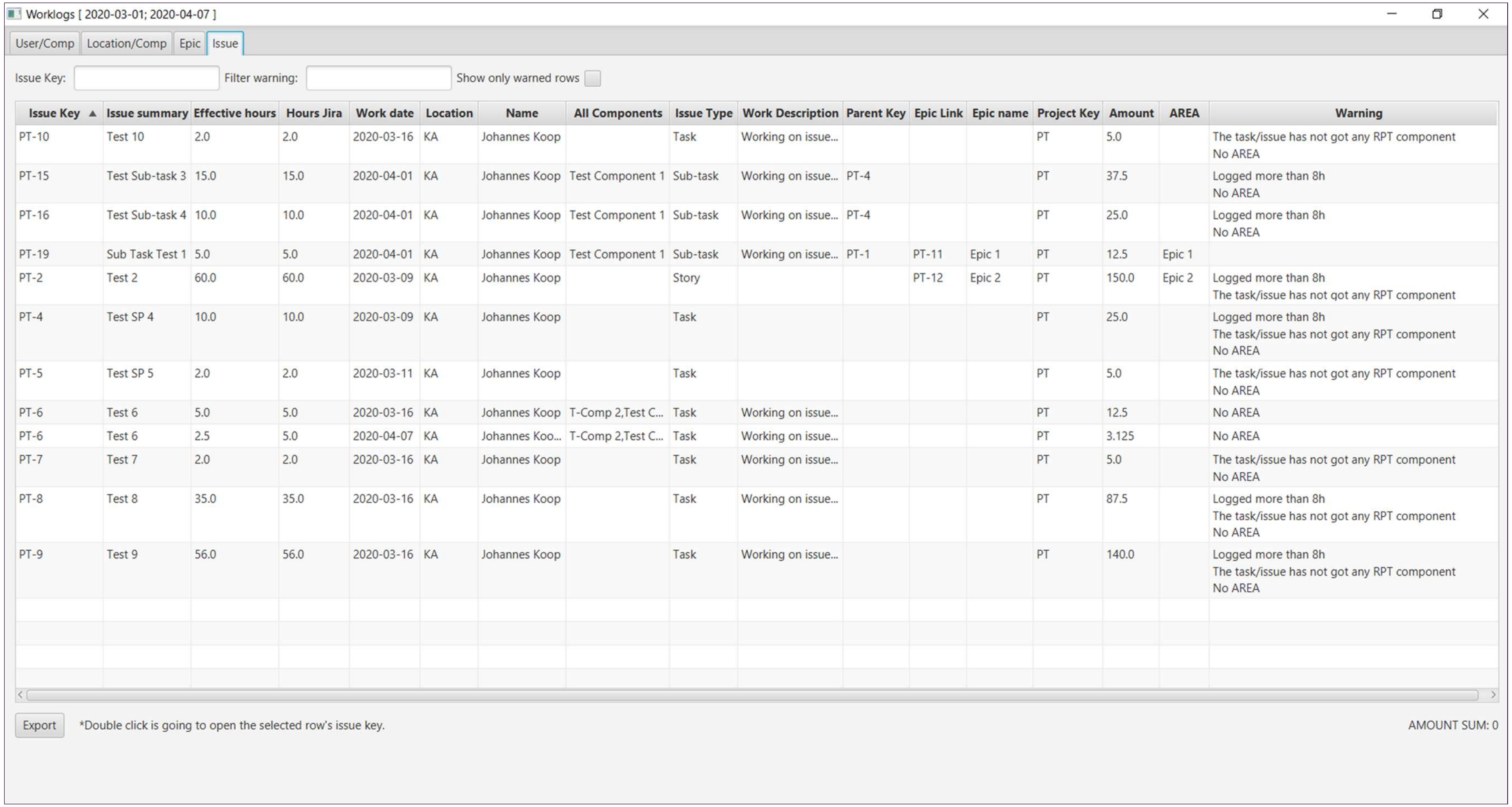
Appendix 16: TimeTrackingTool – Effort Mode - User/Comp Tab



Name	Location	JIRA component	Project Key	Hour/min	Hours	Ext. Effort Hour/min	External effort
Johannes Koop	KA		PT	167h 0m	167.0	167h 0m	167.0
Johannes Koop	KA	T-Comp 2,Test Component 1	PT	20h 0m	20.0	20h 0m	20.0

SUM: 0 hours

Appendix 17 – Effort Mode – Issue Tab



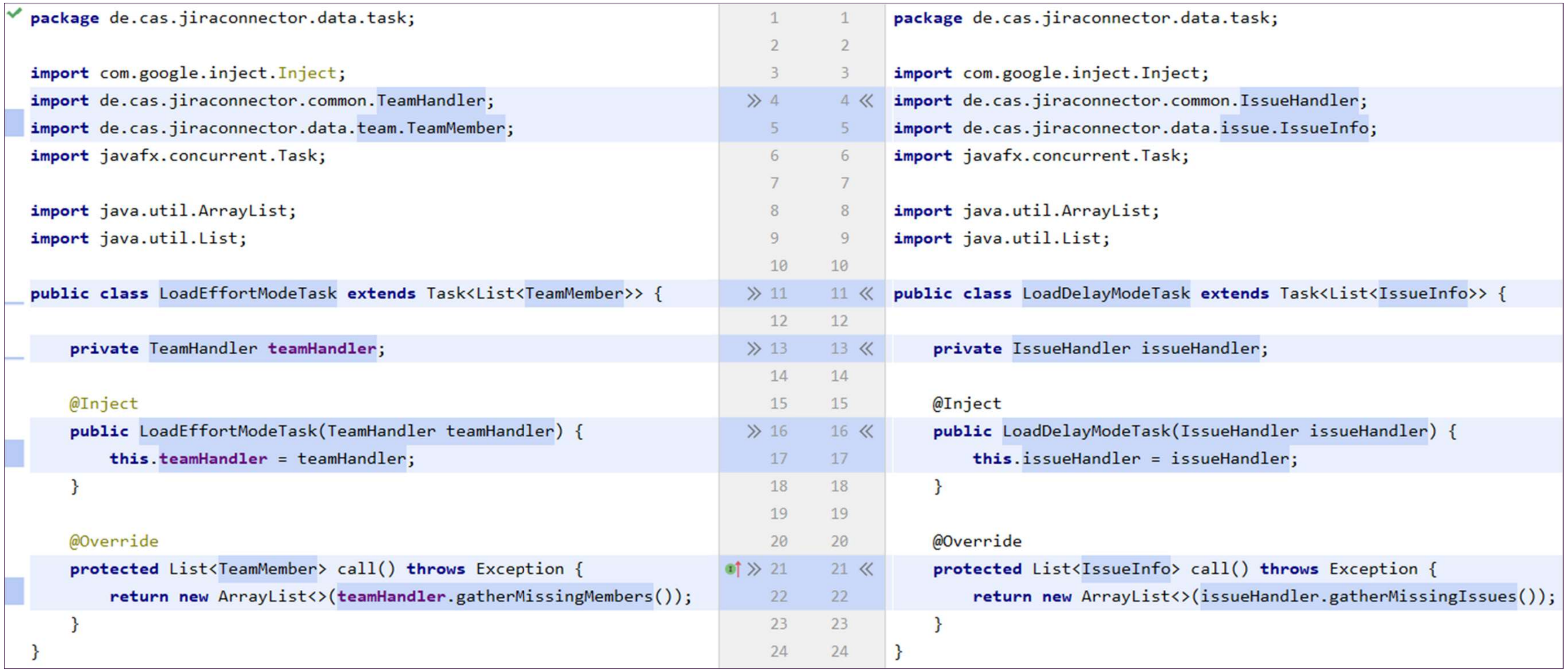
Issue Key: Filter warning: Show only warned rows □

Issue Key ▲	Issue summary	Effective hours	Hours Jira	Work date	Location	Name	All Components	Issue Type	Work Description	Parent Key	Epic Link	Epic name	Project Key	Amount	AREA	Warning
PT-10	Test 10	2.0	2.0	2020-03-16	KA	Johannes Koop		Task	Working on issue...				PT	5.0		The task/issue has not got any RPT component No AREA
PT-15	Test Sub-task 3	15.0	15.0	2020-04-01	KA	Johannes Koop	Test Component 1	Sub-task	Working on issue...	PT-4			PT	37.5		Logged more than 8h No AREA
PT-16	Test Sub-task 4	10.0	10.0	2020-04-01	KA	Johannes Koop	Test Component 1	Sub-task	Working on issue...	PT-4			PT	25.0		Logged more than 8h No AREA
PT-19	Sub Task Test 1	5.0	5.0	2020-04-01	KA	Johannes Koop	Test Component 1	Sub-task	Working on issue...	PT-1	PT-11	Epic 1	PT	12.5	Epic 1	
PT-2	Test 2	60.0	60.0	2020-03-09	KA	Johannes Koop		Story			PT-12	Epic 2	PT	150.0	Epic 2	Logged more than 8h The task/issue has not got any RPT component
PT-4	Test SP 4	10.0	10.0	2020-03-09	KA	Johannes Koop		Task					PT	25.0		Logged more than 8h The task/issue has not got any RPT component No AREA
PT-5	Test SP 5	2.0	2.0	2020-03-11	KA	Johannes Koop		Task					PT	5.0		The task/issue has not got any RPT component No AREA
PT-6	Test 6	5.0	5.0	2020-03-16	KA	Johannes Koop	T-Comp 2,Test C...	Task	Working on issue...				PT	12.5		No AREA
PT-6	Test 6	2.5	5.0	2020-04-07	KA	Johannes Koo...	T-Comp 2,Test C...	Task	Working on issue...				PT	3.125		No AREA
PT-7	Test 7	2.0	2.0	2020-03-16	KA	Johannes Koop		Task	Working on issue...				PT	5.0		The task/issue has not got any RPT component No AREA
PT-8	Test 8	35.0	35.0	2020-03-16	KA	Johannes Koop		Task	Working on issue...				PT	87.5		Logged more than 8h The task/issue has not got any RPT component No AREA
PT-9	Test 9	56.0	56.0	2020-03-16	KA	Johannes Koop		Task	Working on issue...				PT	140.0		Logged more than 8h The task/issue has not got any RPT component No AREA

*Double click is going to open the selected row's issue key.

AMOUNT SUM: 0

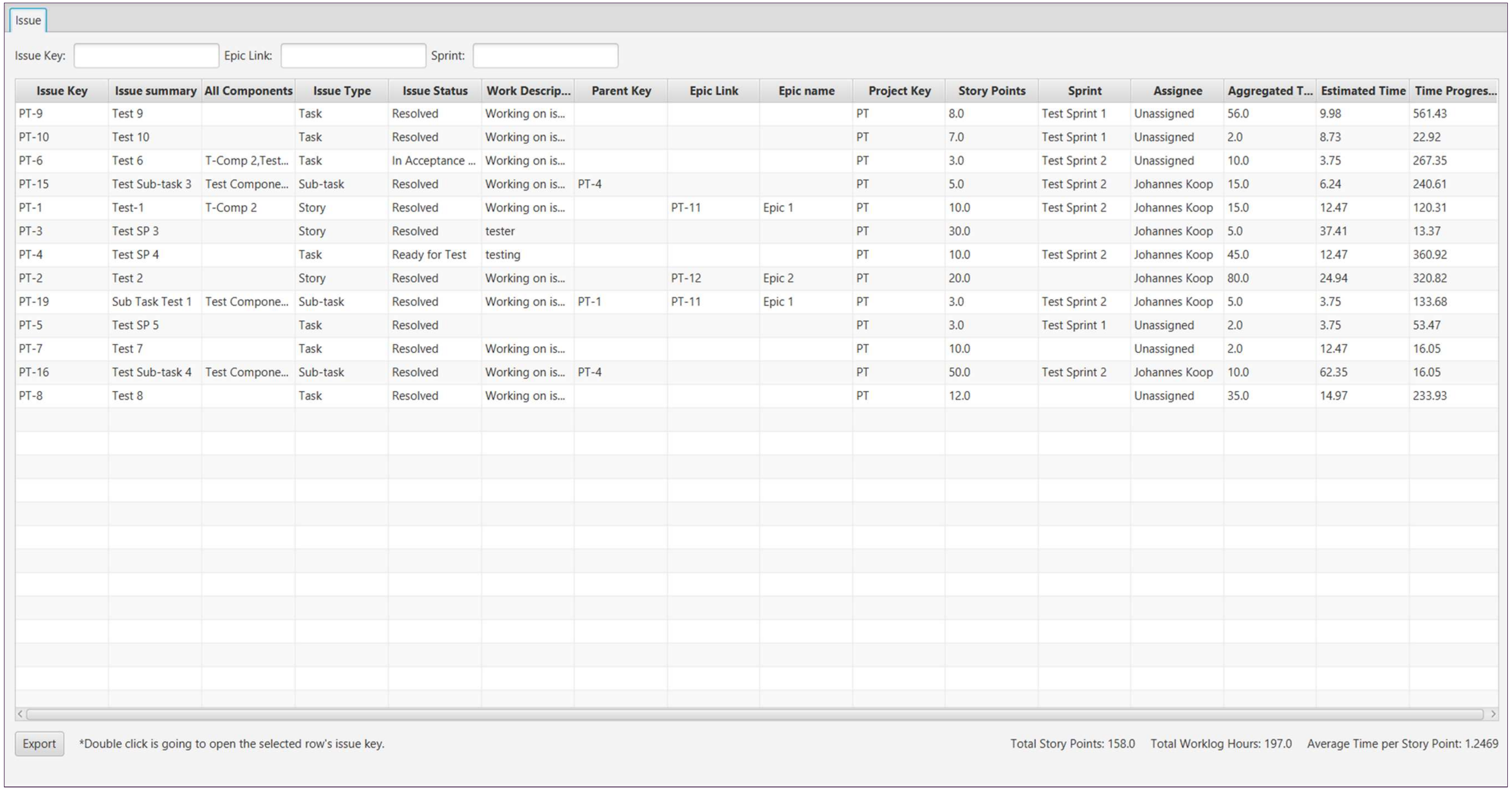
Appendix 18 – LoadEffortModeTask comparison LoadDelayModeTask



```java
package de.cas.jiraconnector.data.task;

import com.google.inject.Inject;
import de.cas.jiraconnector.common.TeamHandler;
import de.cas.jiraconnector.data.team.TeamMember;
import javafx.concurrent.Task;

import java.util.ArrayList;
import java.util.List;

public class LoadEffortModeTask extends Task<List<TeamMember>> {

    private TeamHandler teamHandler;

    @Inject
    public LoadEffortModeTask(TeamHandler teamHandler) {
        this.teamHandler = teamHandler;
    }

    @Override
    protected List<TeamMember> call() throws Exception {
        return new ArrayList<>(teamHandler.gatherMissingMembers());
    }
}
```

```java
package de.cas.jiraconnector.data.task;

import com.google.inject.Inject;
import de.cas.jiraconnector.common.IssueHandler;
import de.cas.jiraconnector.data.issue.IssueInfo;
import javafx.concurrent.Task;

import java.util.ArrayList;
import java.util.List;

public class LoadDelayModeTask extends Task<List<IssueInfo>> {

    private IssueHandler issueHandler;

    @Inject
    public LoadDelayModeTask(IssueHandler issueHandler) {
        this.issueHandler = issueHandler;
    }

    @Override
    protected List<IssueInfo> call() throws Exception {
        return new ArrayList<>(issueHandler.gatherMissingIssues());
    }
}
```

Appendix 19 – Delay Mode



Issue Key	Issue summary	All Components	Issue Type	Issue Status	Work Descrip...	Parent Key	Epic Link	Epic name	Project Key	Story Points	Sprint	Assignee	Aggregated T...	Estimated Time	Time Progres...
PT-9	Test 9		Task	Resolved	Working on is...				PT	8.0	Test Sprint 1	Unassigned	56.0	9.98	561.43
PT-10	Test 10		Task	Resolved	Working on is...				PT	7.0	Test Sprint 1	Unassigned	2.0	8.73	22.92
PT-6	Test 6	T-Comp 2,Test...	Task	In Acceptance ...	Working on is...				PT	3.0	Test Sprint 2	Unassigned	10.0	3.75	267.35
PT-15	Test Sub-task 3	Test Compone...	Sub-task	Resolved	Working on is...	PT-4			PT	5.0	Test Sprint 2	Johannes Koop	15.0	6.24	240.61
PT-1	Test-1	T-Comp 2	Story	Resolved	Working on is...		PT-11	Epic 1	PT	10.0	Test Sprint 2	Johannes Koop	15.0	12.47	120.31
PT-3	Test SP 3		Story	Resolved	tester				PT	30.0		Johannes Koop	5.0	37.41	13.37
PT-4	Test SP 4		Task	Ready for Test	testing				PT	10.0	Test Sprint 2	Johannes Koop	45.0	12.47	360.92
PT-2	Test 2		Story	Resolved	Working on is...		PT-12	Epic 2	PT	20.0		Johannes Koop	80.0	24.94	320.82
PT-19	Sub Task Test 1	Test Compone...	Sub-task	Resolved	Working on is...	PT-1	PT-11	Epic 1	PT	3.0	Test Sprint 2	Johannes Koop	5.0	3.75	133.68
PT-5	Test SP 5		Task	Resolved					PT	3.0	Test Sprint 1	Unassigned	2.0	3.75	53.47
PT-7	Test 7		Task	Resolved	Working on is...				PT	10.0		Unassigned	2.0	12.47	16.05
PT-16	Test Sub-task 4	Test Compone...	Sub-task	Resolved	Working on is...	PT-4			PT	50.0	Test Sprint 2	Johannes Koop	10.0	62.35	16.05
PT-8	Test 8		Task	Resolved	Working on is...				PT	12.0		Unassigned	35.0	14.97	233.93

*Double click is going to open the selected row's issue key.

Total Story Points: 158.0 Total Worklog Hours: 197.0 Average Time per Story Point: 1.2469