UNIVERSITY OF TARTU
Institute of Computer Science
Informatics Curriculum

Karl Toomas Vana

Visualising Jira data for planning accuracy

Bachelor's Thesis (9 ECTS)

Supervisor(s): Alexander Nolte
Ezequiel Scott

Tartu 2021

Visualising Jira data for planning accuracy

Abstract:

The Scaled Agile Framework (SAFe) is commonly used to organise software development in large companies. Jira is a project management tool that helps software development companies to track, plan, and analyse their work processes. Despite its flexibility, Jira lacks the functionality to monitor the planning accuracy of teams when SAFe is used. This thesis proposes a dashboard to help companies using SAFe to assess initial sprint planning accuracy based on the data extracted from Jira. The application can calculate the planning accuracy for teams and break it down to the feature level to give detailed insight into teams plans and outcomes. The dashboard was well received by the test group and simplified their job.

Keywords:

Jira, agile development, Scrum, SAFe, planning accuracy

CERCS: P170

Jira andmete visualiseerimine planeerimistäpsuse hindamiseks

Lühikokkuvõte:

Scaled Agile Framework (SAFe) on laialt kasutatav agiilne arendusmetoodika, mis on mõeldud arendusprotsesside organiseerimiseks suurtes ettevõtetes. Tööprotsesside haldamiseks, planeerimiseks ja analüüsimiseks võib toetuda Jira tarkvarale. Vaatamata Jira suurele paindlikkusele, puudub sellel funktsionaalsus, mis aitaks jälgida tiimide planeerimise täpsust kui kasutatakse SAFe metoodikat. Bakalaureusetöö raames loodud veebileht aitab Jira andmetele põhinedes SAFe metoodikat kasutaval ettevõttel algset sprindi planeerimistäpsust hinnata. Lisaks täpsuse arvutamisele suudab programm anda ülesannete tasandil iga tiimi kohta ülevaate nende algsetest plaanidest ning lõplikust tulemusest. Veeblehe testimisgruppi kuulunud kasutajad olid programmiga rahul ning see lihtsustas nende tööd.

Võtmesõnad:

Jira, agiilne arendus, Scrum, välkarendus, SAFe, planeerimistäpsus

CERCS: P170

Table of Contents

1 Introduction

As the world is changing rapidly, so does the technology and systems that are used. To extend and maintain software at a high pace, the favoured solution would be to use an agile approach, as stated in the *14th State of Agile Report* [1]. Such approaches would be using Scrum[1] and Scaled Agile Framework[2] (SAFe), which propose dividing work into Program Increments (PI) and sprints. SAFe is meant for big companies and large systems to organise and speed up their development processes, as stated by the creator of SAFe Dean Leffingwell [2]. In the given methodology, the PIs are each consisting of usually four development iterations, followed by innovation iteration and planning iteration [3]. Different teams are separated into Agile Release Trains (ART) and are coordinated by a Release Train Engineer (RTE). Some of RTE's responsibilities are locating and lowering risks, solving problems that are stopping development, supervising the PI, and later assessing the PI outcomes [3].

There are many software project management tools available that allow teams to organise and track their work. One of such tools is Jira[3]. In addition to keeping track of issues that are going to be developed, Jira also gives the ability to divide the tasks between sprints, mark estimations on how long the task completion will take and monitor its progress. Jira does not provide functionality for doing planning accuracy assessment. Accuracy is calculated by comparing the times when team planned to deliver a feature and when they actually delivered it. Currently, RTE must collect all the needed data manually from Jira to analyse the planning accuracy. Which is used to evaluate how well the teams are planning their work, how reliable are their plans, offer them support if needed, and avoid mistakes made in previous plannings.

This thesis aims to design, create, and test a solution that supports RTEs in assessing the planning accuracy. The solution calculates the planning accuracies for teams, clearly visualises the related data, and reduces the manual work needed.

The thesis structure is as follows: problem introduction, describing the background, methodology used to build the solution, results, solution architecture and conclusions.

[1] https://www.scrum.org/resources/what-is-scrum
[2] https://www.scaledagileframework.com/
[3] https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for#Jira-for-requirements-&-test-case-management

2 Background

The author of the thesis works at Proekspert[4] as a quality engineer. The problem originated from a client who is a big corporation and uses SAFe to coordinate their work. The problem arises from the needs of the Release Train Manager and the limitations of Jira.

The project was led by the author of the thesis. That included analysing the problem, meetings with the client to discuss the progress and next steps, planning the solution, programming, and coordinating user testing. None of the tasks is part of quality engineers everyday work.

2.1 SAFe

Scaled Agile Framework (SAFe) is a collection of proven and integrated principles, practices, and knowledge for agile operating large-scale enterprises [4]. SAFe offers competencies for achieving business agility, responding quickly to changing customer needs and emerging new technologies [4]. For organising the development, the teams are divided into Agile Release Trains which simplifies the development planning and communication between teams and stakeholders [4]. The trains work is divided into sprints that are 2-3 weeks long [3]. The planning of the sprints is coordinated by Release Train Engineer [5]

2.2 Planning accuracy

The problem of assessing teams' planning accuracy originates from the Release Train Engineer's needs (RTE). The purpose of an RTE is to coordinate Agile Release Train (ART) processes, which include communicating with stakeholders, solving upcoming problems, managing the risks, facilitating PI planning, and later tracking the progress of PI, which includes the planning accuracy of teams [5]. If the RTE wants to assess how accurately the teams are planning their sprints, then the teams must fill in a custom field on each issue that states in which sprint they are planning to finish the feature. The content of that field will later be compared to the actual date the task was completed. According to Scaled Agile Framework guidelines, percentage of accomplished release objectives is one of the metrics used to evaluate the train's performance and progress [6]. It is calculated by comparing the story points and the actually delivered points per feature which is also referred to as release predictability [6]. The metric helps to understand how predictable the train's progress is and where changes might be needed. This also gives input to stakeholders on how reliant the train is. It is also brought out that the teams release predictability, the per cent of objectives achieved, should remain between 80-100% [6]. It states that when a team is constantly operating at 100%, then this means the team is under-planning or not taking any risks and innovation [6]. The planning accuracy in the thesis compares the number of features planned and the number of features delivered according to the initial plan, to present the same reliability as when comparing objectives by story points planned and delivered.

2.2.1 Limitations of Jira when assessing planning accuracy

Custom fields are added to issues in Jira to track and plan the PIs better. One of those fields is used to predict at which sprint the feature will be ready. Some of the features might not be completed on time because of some complications or business decisions. For tracking the current status of the issue, another field is used.

At the end of PI, Jira offers different graphs and summaries for the PI, but they have a shortcoming. No summary would take into account the custom fields created by RTE. For example, it is possible to see that from the 20 issues planned to the sprint, only 18 were

[4] https://proekspert.com/

completed, but it does not mean that the 18 issues were exactly the same ones that were planned initially. In addition to moving tasks between sprints, adding new tasks to the sprint is possible if needed. Tracking these changes is complicated because Jira does not support searching for changes in the past for custom fields. This means that it is impossible to search whether there was a change in the custom field value between specific dates in the past.

Jira's development teams at Atlassian are aware of the need for such functionality as users have requested it, but they are not planning to take it into development [7]. As a result, the data must be gathered manually to assess the PI planning accuracy. This process can take three or more hours, according to the RTE of the client. The process for collecting data manually goes as follows: check all the tasks teams planned to deliver and in which sprint they planned to complete it, go through all the tasks to see if they were finished on promised time and calculate the planning accuracy for each team. Although the data can be checked manually, there is no way to keep track of any upcoming changes and the whole process would have to be repeated.

2.2.2 Existing Jira plugins

A couple of solutions could help with finding planning accuracy, but all of them have their shortcomings, and none of them is made for this sole purpose. One of the most common problems is that existing plugins that offer a way to visualise or analyse Jira history data do not support custom labels and often focus on a specific field like status, description, or time-tracking. This section will bring out and analyse the existing history analysing plugins that do support custom fields.

Issue History Jira plugin

Given plugin offers the possibility to see all the changes made to an issue in a given time range, supporting advanced search functions to select issues by project or any label [8]. This plugin, therefore, solves the problem of searching changes made to a custom field between specific dates. However, if used to find planning accuracy, then the user would have to still go through all the issues of interest by hand and write down the changes made in PI. Another problem would be that this plugin is only meant to be used on Jira cloud instances, but the thesis author's client uses only on-premise Jira servers.

Issue History Timeline plugin

This plugin offers the users the possibility to compare changes made to an issue between specific dates and filtering by custom fields [9]. The plugin seems to put the main emphasis on visualising the Jira changelog. When assessing the planning accuracy, it would be helpful to filter out changes made to a custom field between given dates, but this would still not eliminate the need to go through all the issues one by one to extract the data.

3 Methodology

To understand the scope of the problem, priorities, possible solutions and guiding development, multiple meetings (figure 1) were held between the client and the author of the thesis. The discussions included setting priorities for the solution, presenting wire-frames, and doing demos once a part of the functionality was completed.

Before the first meetings between the author of the thesis and the client, it was already known that putting together reports after the PI end is causing problems for the client. A hackathon was held by Proekspert to get started on solving different problems the client had proposed. The employees were divided into teams and given various topics to work on. The author of the thesis was in a group of 5 working on the problem described as follows: Putting together PI reports is complicated and requires much manual work. As the initial description of the topic was quite vague, then a considerable amount of time went into identifying the actual problems. The hackathon's outcome was research on different possibilities of generating custom reports based on Jira's data and trying out the Jira REST API. It was also agreed that the author of the thesis would be the one who will continue moving towards a real solution.

Following the meetings, preparations, discussions and outcomes are described in more detail.

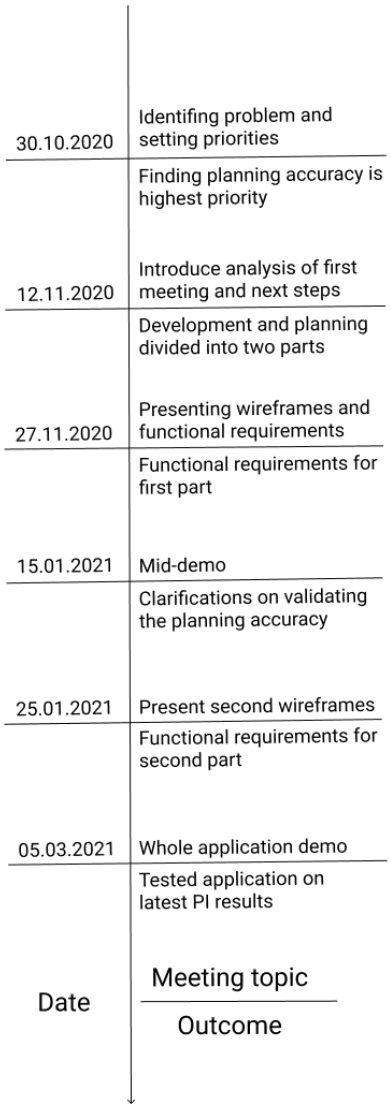


Figure 1. Client meetings timeline and topics.

First meeting

Before the first meeting, the author of the thesis had prepared an agenda and set the goals for the meeting. The agenda was planned as follows:

1. Introduction who is the author of the thesis and what is the aim of the meeting
2. What info does the author already have and are these the real needs?
3. Go through the process client wants to simplify/automate
4. Find the biggest pain points and set the priorities
5. General questions
6. Propose next actions and set next meeting

The goals set for the meeting were:

- Understand what is the biggest problem and also what are the things that are already working?
- Set priorities for the features that are needed.
- Agree on what would be the minimum viable product (MVP).
- Write down the specification for needed features of the solution. That should also include the related information in Jira, what tags to consider etc.
- Set a date for the next meeting to propose a timeline and a way for resolving the problem.

The first meeting was held on 30.10.2020 and the participants were the author of the thesis, Release Train Engineer and Product Development Manager (PDM) from the client side and Product Owner and Key Account Manager from Proekspert. The input got from the client before the meeting listed several following aspects that were taken into consideration when putting together the PI report:

- Committed features based on teams
- Modifications in target sprints, feature types etc.
- Added/rejected feature in PI
- Ready for UAT dates
- Percentage of features that are completed
- The team following their PI plan most accurately
- The number of defects introduced during the development/UAT
- Comparison by weeks, sprints and PIs
- How much time was actually spent on features and how it differed from estimates

Firstly, all the steps listed above were discussed one by one. The RTE would like to be able to see the committed features based on teams when assessing the planning accuracy. It was agreed by both RTE and PDM that the biggest problem is tracking the changes done in the *Target sprint* custom field that is used to mark down the sprint where the team plans to finish the task. Later the same field can be changed if, for some reason, the team cannot deliver according to the initial plan. There is no automatic way to distinguish and track such changes in Jira and in case the changes have been made after the RTE has already checked the data, then there is no way to get information that a change has occurred. In addition, the RTE would like to see added and rejected features to PI, the date when features were marked as *Ready For UAT*, percentage of features that are completed according to the initial plan, also referred to as planning accuracy. In addition they would like to be able to compare the teams based on their planning accuracy. The second biggest problem is comparing the difference between how much time the team spent on a feature and how much they planned to spend. As the second problem is quite different from the first one, it was agreed that the author of the thesis would work on finding a

solution to the planning accuracy problem only. After discussing the previously known points, RTE described the work needed to get planning accuracy manually. The steps are:

1. Find all features that teams are committed to in PI and mark them down in Excel
2. Find all initial target sprints for the features
3. If time, update the excel at the end of every sprint
4. At the end of sprint 5, find all the dates when the status of the issue was marked as *Ready for UAT*
5. Mark down if some features have been removed or rejected from sprint for outside reasons and tasks that are inherently just supporting testing for some other team and not delivering a feature
6. Count how many features teams planned and how many they delivered on time to get the planning accuracy
7. Compare the figures to find the team with the best planning accuracy

The meeting was ended with a round of questions from the author of the thesis. It was clarified that putting together the PI reports containing planning accuracy manually can take three to five hours. The process has never been tried to be simplified before and it is possible that RTEs from other Agile Release Trains would also be interested in the solution.

The meeting met four of the goals set for the meeting. The biggest problem is comparing the changes done to the *Target sprint* field that is used to assess the planning accuracy. This was also set as the problem that the upcoming project should solve to be an MVP. There was not enough time to go through all the feature descriptions that the project should have. The next meeting was scheduled to be after two weeks.

Second meeting

Before the second meeting, the author of the thesis had confirmed that, indeed, searching over history data of custom fields cannot be done in Jira. Which means that it is not possible to retrieve all issues where has been a change in a custom field between certain dates, as pointed out by the client in the first meeting. He also investigated Jira REST API's possibilities and that all the needed data can be retrieved from there. After that, he compared different ways of making the interface. Options were to make a Jira dashboard item[5], make a separate Jira subpage using AUI components[6] that are the ones Jira itself uses or make an entirely separate webpage with Angular and Typescript. The author of the thesis decided the best way to go would be to make a Jira dashboard item so all the data and interface would be in one place. The agenda for the meeting was set as follows:

1. Recap from the first meeting
2. Solution idea
3. Next steps

The second meeting was held on 12.11.2020 between the same people as in the first meeting. The aim of the second meeting was to introduce the analysis done after the first meeting and agree on the next steps.

After the recap from the first meeting, the author of the thesis presented a solution idea to make a Jira dashboard item. He pointed out that the Jira dashboard item would be the most convenient solution so all the things could be kept in one place. The author of the thesis had also discussed it with the managers of the clients on-premise Jira instance and confirmed that such a solution

[5] https://developer.atlassian.com/server/jira/platform/dashboard-item-module/

[6] https://aui.atlassian.com/aui/latest/

could be implemented. He also showed how the dashboard items could be accessed and used from Jira. Following, the author of the thesis proposed that the next step would be to divide the application into two parts. The first part would be displaying by teams which features they completed that were promised initially and which tasks had a change in the *Target sprint* field. He also proposed that the next meeting would have the aim to show the wire-frame for the first part of the page and gather feedback. RTE responded to the proposal as "At least I don't have any questions, I'm happy with the proposal to have a dashboard". The time for the next meeting was not set yet as it wasn't clear how long making the wire-frames would take.

Third meeting

For the third meeting, the author of the thesis had prepared a wire-frame for the first part of the page as planned in the previous meeting. For preparing the wire-frame, the author of the thesis made a list of functional requirements (see results section) for one team issues view and for making the request. Writing the functional requirements was based on the description of manual steps needed to assess planning accuracy that was described by the client in the first meeting.

The meeting was held to introduce the wire-frames for agreeing on functionality. It was held on 27.11.2020 between the author of the thesis and the RTE of the client. The questions the author was looking for answers to while showing the prototype were:

1. Does the client understand how the prototype works?
2. In what order would the user do the steps when looking for planning accuracy of one team?
3. What data is the user looking for as the first thing?
4. If a user imagines using the prototype, then what problems could he/she face?
5. What is missing from the prototype?
6. What would the user change about the prototype?
7. Is there anything unneeded within the prototype?
8. Can the sprint dates be automatically found from Jira?

The wire-frame (figure 2) was sent to the customer.

Figure 2. Wire-frame for the issues table and search inputs.

The first question asked while the client was examining the prototype was whether the client understands how the prototype works. The client responded by thinking out loud, "There are the sprints, then I can check the team and all the issues, issues with change in sprint, initial and revised targets. Yes, it looks nice". When asked to list the steps to get the planning accuracy for one team based on the prototype then the client imagined using it as follows:

1. First choose one team
2. Then check all the issues one-by-one and see which ones have a change in target sprint

The client could not point out anything in particular that she is looking for. Also, the client said that the prototype looks nice and she cannot imagine any problems when using the prototype. When asked if there is anything unneeded then the client pointed out that all the fields in the table are important information and that she wouldn't exclude anything from the prototype. The last question was if the sprint dates can be automatically retrieved from Jira? The client stated that, unfortunately, there isn't a way to do that, but entering them manually would not be a problem, only a minor issue. The author of the thesis updated the requirements to include a functionality where all dates would be calculated automatically after entering the start date of the first sprint.

As the client understood the prototype and was happy with the presented functionality then the functional requirements set for making the request and issues table were correct. The topic for the next meeting was set to be a demonstration of the work done for the first part of the page.

Fourth meeting

By the time of the fourth meeting, the development progress was not too significant due to many problems that the author faced when developing the Jira dashboard item. The biggest problem was getting the local development environment to work with Atlassian JDK. Many of the setup problems got solved, but the custom dashboard item could not be loaded. Also, a request was made to Atlassian support, who said they do not know how to solve the problem and suggested looking into Atlassian developer forums. The forums mainly had outdated posts and people looking to solve the same problems but no solutions. After three weeks of trying, the author of the thesis found it feasible to continue with another approach. The next idea was to make an Angular page that could be displayed as a Jira subpage but at the same time could be run separately from Jira. After trying out such a solution, it became clear that it would be considerably faster to develop the page if there is no association to Jira other than using Jira API. The agenda prepared for the meeting was following:

1. Progress and problems overview
2. New approach introduction
3. Status of the project and demo
4. What are the next steps?

The aim of the fourth meeting was to talk about the progress, status of the page, demo the first part of the page and gather feedback. The meeting was held on 15.01.2021 between the author of the thesis, RTE from the client side, Product Owner and Key Account Manager from Proekspert. Firstly, the author of the thesis described the problems that he had faced while developing the Jira dashboard item. Secondly, the new approach of using Angular, Typescript and Angular Material components was introduced to the client. It was pointed out by the author of the thesis that using a different approach would be faster to develop, could be run separately from Jira, but would also leave open the possibility to display it as a Jira subpage. The progress overview was followed by a demo of the first part of the page. At that moment, the implemented functionalities were PI date selecting, PI label input, team selecting, fetching data from API, parsing Jira issues history and displaying the data in a table. The demo bought out multiple improvement needs. Firstly, the client pointed out that some issues are missing from the table, which later turned out to be a bug. Another improvement needed was to loosen the validation of whether the issue was completed on time or not. The reason was that usually, the teams have their sprint plannings on the following Monday after the sprint ends and that is the time when they will update the issues in Jira. The client could not point out anything missing or needed to be added to the page part of the application that was demonstrated.

The author of the thesis proposed the next steps to be improvements to the first part of the application, agree on logic to estimate the planning accuracy, present wire-frames of the second half of the page, second demo, launch and user testing. The logic for planning accuracy was discussed straight after. The client pointed out that testing support tasks should be excluded from the planning accuracy estimating. The reasoning was, “I’m not counting test support features as those are completely dependent on some other teams work. If one team is not ready, then the second team cannot provide test support in the planned sprint”.

The next meeting was planned to be in a week and the topic would be presenting wire-frames for planning accuracy sections for one team view and all teams view, which is also considered as the second half of the page.

Fifth meeting

For the fifth meeting, the author of the thesis had prepared a wire-frame for showing one team and all teams planning accuracy and related data. For making the wire-frames, the author of the thesis first wrote the functional requirements for corresponding sections based on the first and fourth meeting.

The fifth meeting was held on 25.01.2021 to show wire-frames for the second part of the page. Participants were the author of the thesis and RTE. The second wire-frames focused on dealing with and presenting the planning accuracy itself and related issue counts. A table was added below the issues table (figure 3). The table contains the counts for total planned issues, rejected or removed issues, test support issues, issues delivered by plan and issues not delivered by the plan.

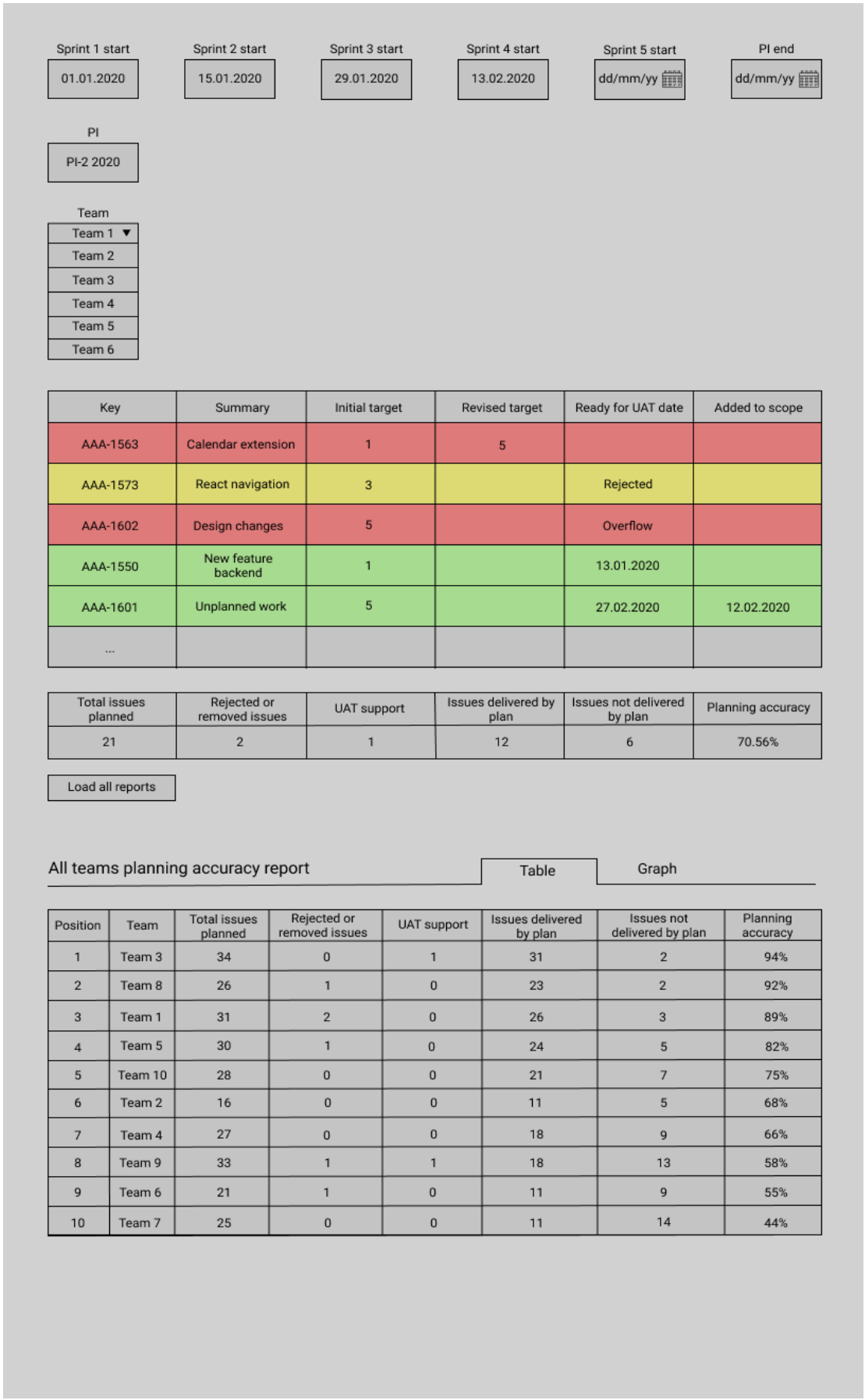


Figure 3. Wire-frame for the teams planning accuracy table.

The author of the thesis asked the client if the presented way of displaying planning accuracy is correct and got an affirmative answer. That also confirmed that the client and author of the thesis had the same functional requirements in mind. The client said the wire-frames look great. The author of the thesis had already made improvements to the first part of the page that was pointed out at the previous meeting, so he also made a demo about the changes. Fixing the bug of not finding all the issues that should be in the PI brought out another problem. There seemed to be too many tasks that had the corresponding PI label. It turned out that in addition to the PI label value, there is another Jira field that has to be taken into account when deciding whether the feature is planned to be done in PI or not. There is a *GO/NO GO* field that is used to mark the final approval for a feature to be taken into the PI. No other questions arose.

Sixth meeting

Before the sixth meeting, the author of the thesis had also finished the implementation of the second part of the page. The aim of the meeting was to demo the second part of the page and obtain the planning accuracy for all the teams for the PI that had just ended at that time

The meeting was held on 05.03.2021 between the same participants as the fourth meeting mentioned earlier. In addition to the features presented at the last demo, the added functionalities were planning accuracy table for one team, planning accuracy summary table for all teams, possibility to see teams as a list sorted by planning accuracy and smaller improvements. The demo revealed a problem where some features were marked as not delivered according to plan, although it seemed they had been completed as planned. The reason was that in some cases, the *Target sprint* field was changed to something else as initially planned and then changed back, but the program identified that as not delivered as scheduled. Other than that, the client was pleased with the solution and the results.

4 Results

Throughout the client meetings, the clients' problems and priorities were discussed, functional requirements were set, a solution proposed and improved gradually.

As an outcome of the first meeting, it was clarified that the biggest problem for the client was assessing the planning accuracy, which is one of the metrics used to evaluate the train's performance and progress [6] in SAFe methodology. That also set the guideline for the minimum viable product, and it should be able to assess planning accuracy and detect issues that had a change in the *Target sprint* field. The meeting also gave a general understanding of what are the steps needed to take to assess planning accuracy manually, which was taken as a basis for setting functional requirements.

In the second meeting, the planning and development process was divided into two sections, the first one being searching and displaying all the issues in PI per one team and the second part displaying the planning accuracy and related data for one team or all teams.

For the third meeting author of the thesis had made a wire-frame for the first part of the page. For making the wire-frame, he had defined the following functional requirements for team view based on the first meeting results:

R1) Making the request
- R1.1) The user must be able to set the dates for the sprints in PI.
- R1.2) The user must be able to enter the PI label.
- R1.3) The user must be able to select a team.

R2) Issues table
- R2.1) The user must be able to see all the features for the selected team and PI.
- R2.2) The user must be able to see which features were rejected.
- R2.3) The user must be able to see which features were removed or added to PI.
- R2.4) The user must be able to see which features had an increase in the T*arget sprint* field.
- R2.5) The user must be able to see which features were testing support tasks.
- R2.6) The user must be able to see which features were not delivered according to the initial plan.
- R2.7) The user must be able to see which features were delivered according to the initial plan.

The client was pleased with the demo and thus the requirements set. The client also brought out that the PI dates cannot be retrieved automatically from Jira. The client didn't consider entering them manually a big problem, but to ease the process, the author of the thesis updated the requirement R.1.1 by adding that some of the dates should be filled manually. The final requirement was

R1.1) The user must be able to set the dates for the sprints in PI. After selecting the first date, the next ones will be filled automatically, but they can be edited.

By the fourth meeting, the author of the thesis has developed the first part of the application. During the demo, it was pointed out by the client that some of the issues are missing that should not have been, which later turned out to be a bug in extracting the right issues from the history data received from Jira. Another improvement requested by the client was to loosen the validation of whether the task was finished on time by adding some extra days. This was a problem since the sprints end on Fridays, but often the data in Jira is updated on the following Mondays. It was also clarified that the client would not like to take testing support tasks into account when calculating the planning accuracy. The reasoning behind this was that the teams planning testing support have no control over the other team and shouldn't be accountable if the other team changes plans or is not delivering accordingly.

For the fifth meeting, the author of the thesis had set the following functional requirements and built a wire-frame for the second part of the page upon them:

R3) Team accuracy table

R3.1) The user must see how many features were planned for the team.

R3.2) The user must be able to see how many features were rejected or removed.

R3.2) The user must be able to see how many features were test support tasks.

R3.3) The user must be able to see how many features were not delivered according to the initial plan.

R3.4) The user must be able to see how many features were delivered according to the initial plan.

R3.5) The user must be able to see the planning accuracy for the team.

R4) Teams summary table

R4.1) The user must be able to see the same data as in the team accuracy table but for all the teams in one table.

R5) Teams summary list

R5.1) The user must be able to see all the teams in a list sorted by their planning accuracy.

The client was pleased with the wire-frames and confirmed that the information displayed in the accuracy table is what was needed. The author of the thesis had also fixed a bug that was brought out in the previous meeting. This resulted in too many issues being displayed, so the author of the thesis went over them with the client once more. It was discovered that the author of the thesis was no aware that there is another field labelled *GO/NO GO* that is used when deciding which features are accepted into the PI.

For the sixth meeting, the author of the thesis had also finished implementing the second part of the page by adding team accuracy table, teams summary table and teams summary list elements. He had also fixed the bug found at the previous meeting. At the sixth meeting, the author of the thesis used the latest PI results to demo the application. The client was happy with the solution. However, another bug was found during the demo. In some cases, there were issues that showed to have a change in the *Target sprint* field, although they didn't. After investigating, it became clear that in some cases, the target sprint was changed to some other sprint and then back, which caused a problem for the application. The bug was fixed right after the meeting.

4.1 User flow

When the user first opens the planning accuracy page, he/she will be redirected to the login screen, which has a button labelled *Login*. By pressing the button, the user is taken to the Auth0 login screen, where the user can fill in their email address and password for authentication. Auth0 also offers an option to recover a password in case the user has forgotten it.

After successful authentication, the user is redirected to the application's main screen (figure 4). On the left, there are options to enter the PI label (requirement R1.2), fill in the dates for the sprints (requirement R1.1), select a team (requirement R1.3) or read the legend for the issues table, which at that moment is not presented to the user yet. On the right user can load all teams' data which will not be available before the user fills in PI and sprint dates. If he/she tries to load all teams' data before that, then corresponding error messages will be shown.

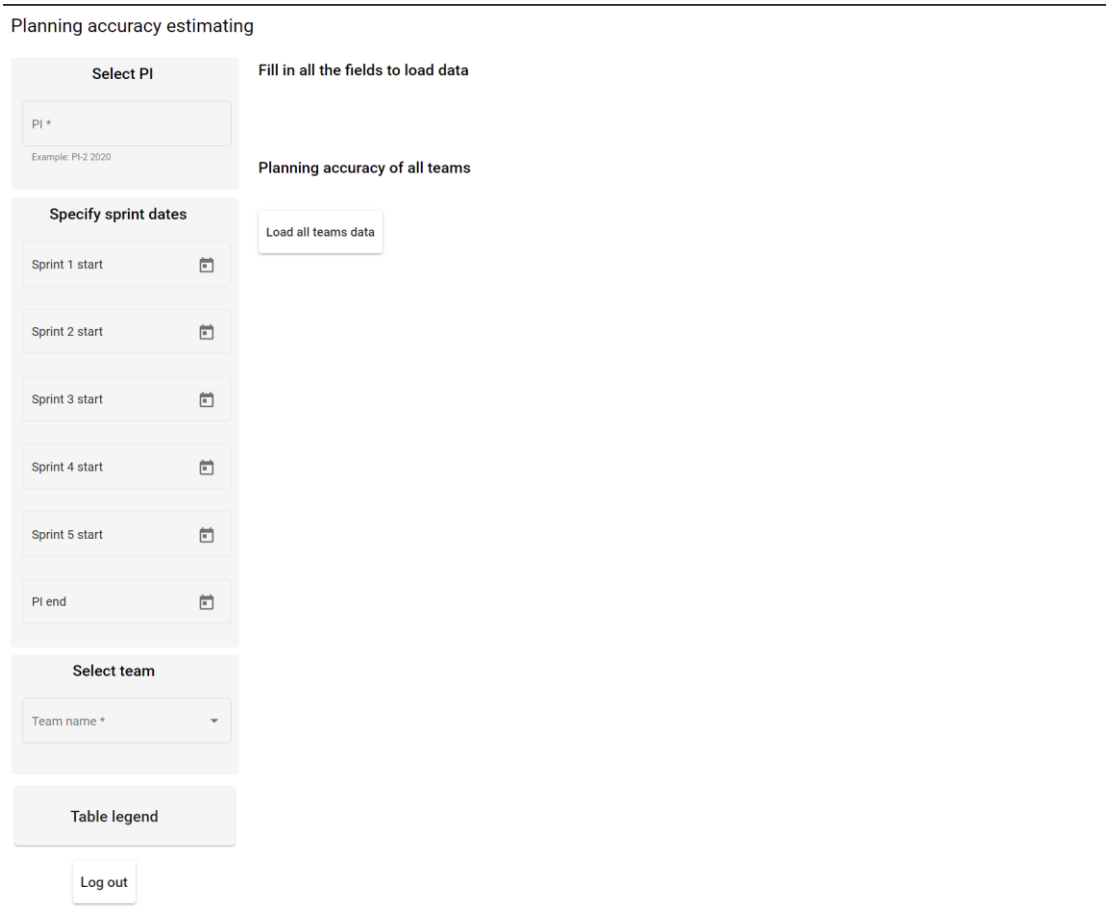


Figure 4. The main screen of the planning accuracy page.

The next step is to fill in the correct label for the PI. The field is an input field that only validates the input as valid if it is entered in the following form *PI-1 2020*. In case the input does not match the pattern, then an error message is shown.

The next step would be to fill in the sprint start dates and the PI end date. The dates can be selected only using a calendar to avoid incorrect inputs. After choosing the first date, the following dates will be filled in automatically with suggestive dates. Usually, the first four sprints are two weeks and the last sprint is three weeks. Also, the sprints do not start on weekends but on the following Monday. If the dates are not correct, then the user has the possibility to correct them one by one.

After the PI label and sprint dates have been entered, then the user can either load the more specific planning accuracy data for one team (requirement R2) or for all the teams combined (requirements R4 and R5). If the user selected one team from the drop-down on the left, then a table with issues and planning accuracy will be loaded on the right (figure 5). The issues table contains all the features that had been planned for the PI (requirement R2.1), the sprint they were scheduled to be finished in and it can be identified by colour if the task was completed according to the initial plan or not (requirement R2.6).

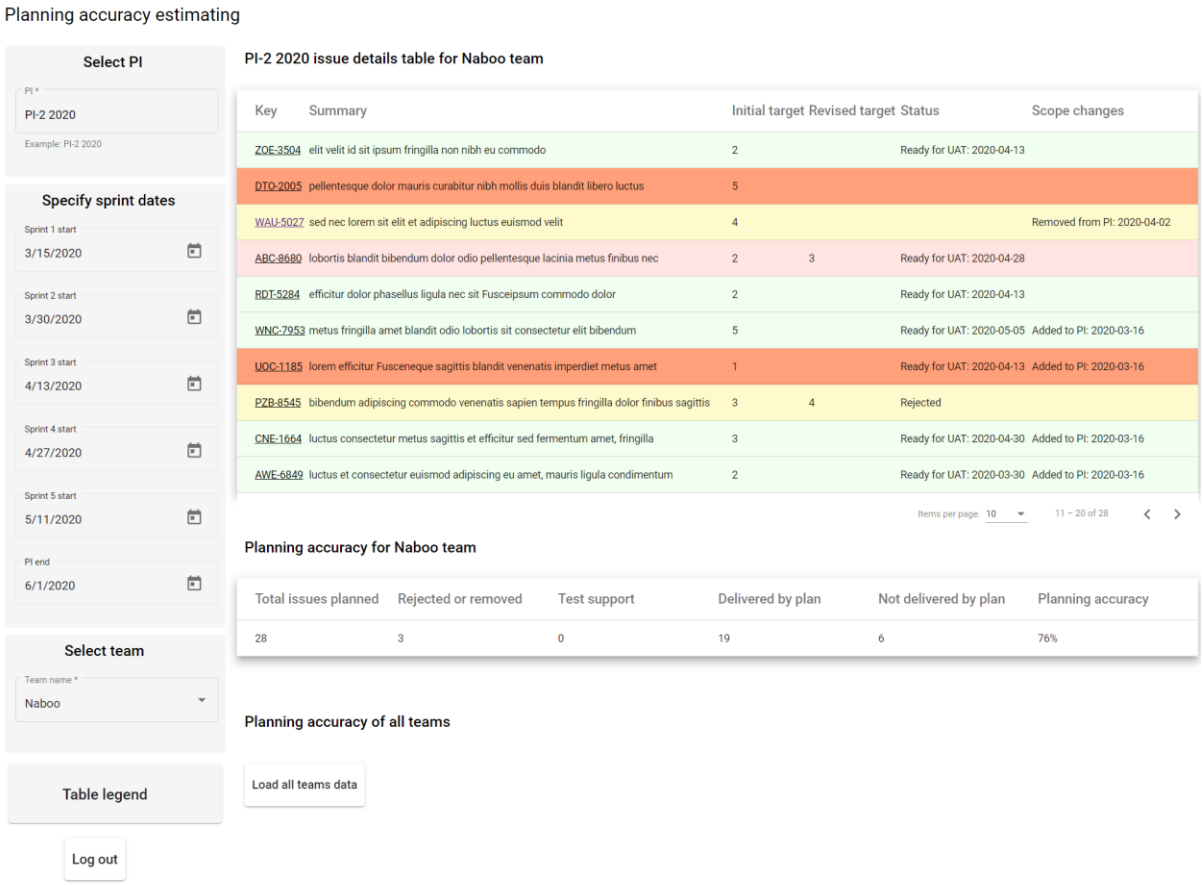


Figure 5. Issues table that is loaded for the selected team.

If the user requires explanations for the table, there is an option to open a legend from the left (figure 6).

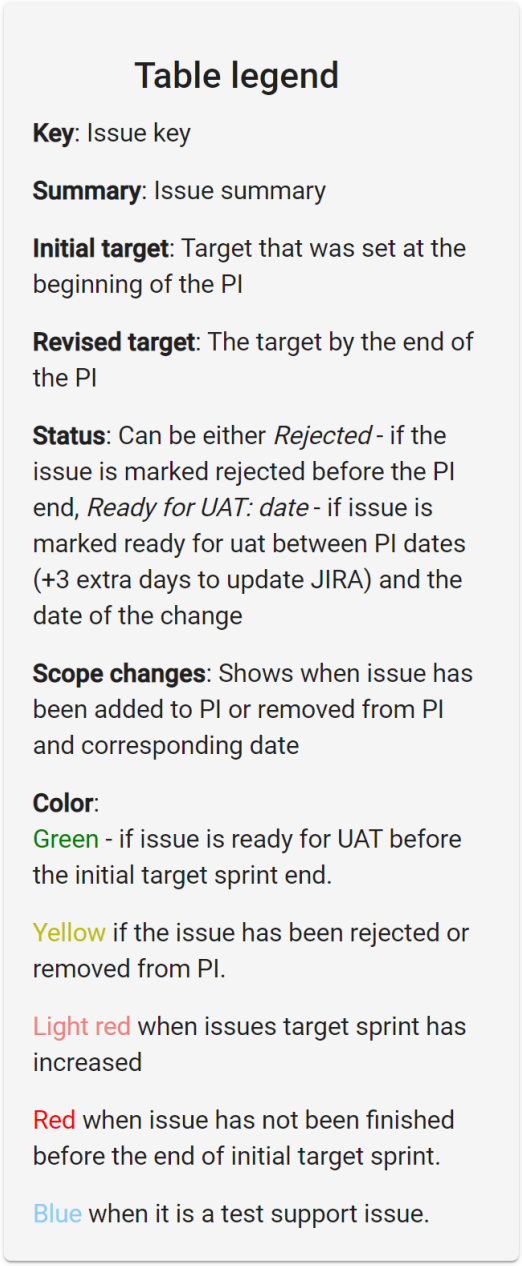


Figure 6. Issues table legend.

If the user clicks on the load all teams' data button, then a loading bar will be displayed (figure 7). The bar moves for every loaded team.

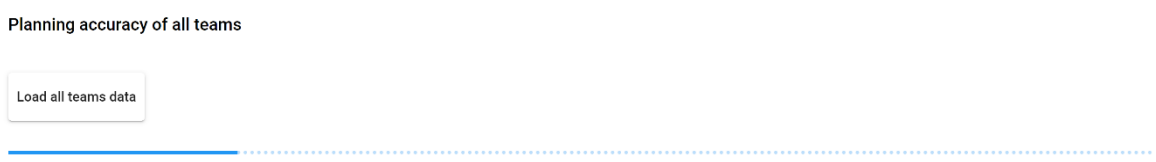


Figure 7. Loading bar while all teams' data is being requested.

Once the loading ends, a list view will open to display all the teams sorted by their planning accuracy (figure 8 and requirement R5.1). The list items will come into display one-by-one using an animation.

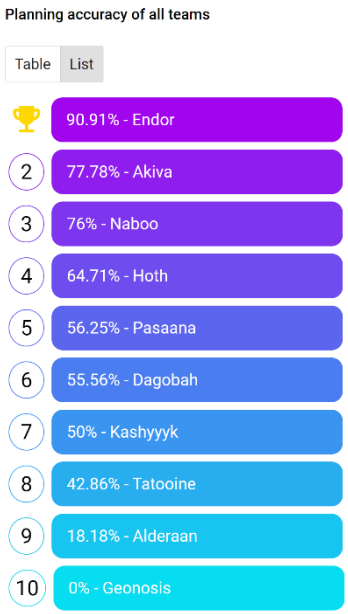
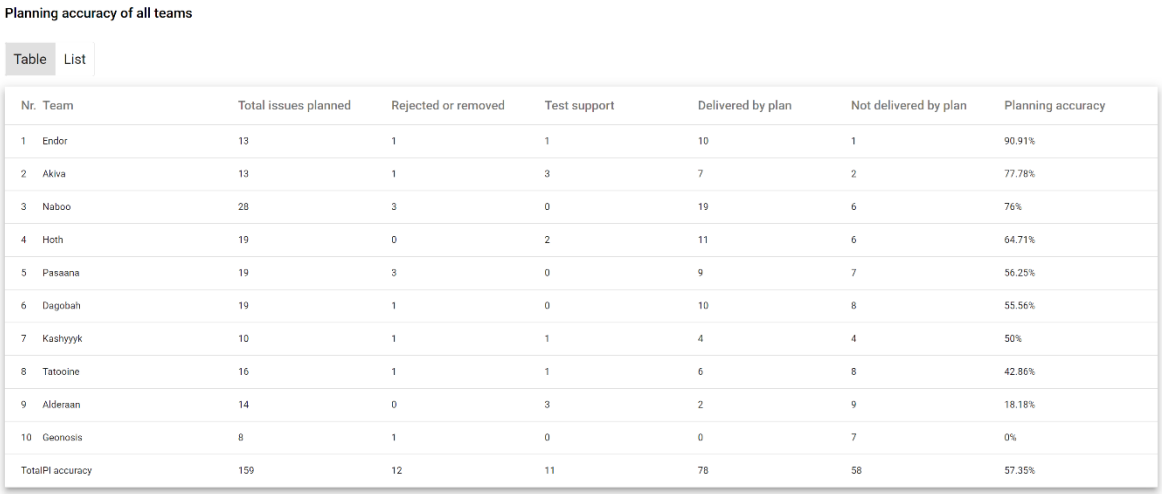


Figure 8. List view of all teams planning accuracies.

The table view (requirement R4.1) will include for every team the total issues planned, the number of issues rejected or removed, the number of test support tasks, the number of issues not delivered according to the initial plan, the number of issues delivered according to plan and also the planning accuracy (figure 9).



Nr.	Team	Total issues planned	Rejected or removed	Test support	Delivered by plan	Not delivered by plan	Planning accuracy
1	Endor	13	1	1	10	1	90.91%
2	Akiva	13	1	3	7	2	77.78%
3	Naboo	28	3	0	19	6	76%
4	Hoth	19	0	2	11	6	64.71%
5	Pasaana	19	3	0	9	7	56.25%
6	Dagobah	19	1	0	10	8	55.56%
7	Kashyyyk	10	1	1	4	4	50%
8	Tatooine	16	1	1	6	8	42.86%
9	Alderaan	14	0	3	2	9	18.18%
10	Geonosis	8	1	0	0	7	0%
TotalPI accuracy		159	12	11	78	58	57.35%

Figure 9. Table view of all teams planning accuracies.

In addition, the table will include a summary of all teams.

4.2 Evaluation

Four user testing sessions were carried out in addition to manual testing and evaluation done by the client during the meetings. The purpose of the testing sessions was to identify the aspects that could be improved within the Planning Accuracy page and also to validate if the solution works as expected. The participants were three Product Owners from Proekspert and RTE from the client side. Users were given a list of tasks (see Appendix 1) and while fulfilling the tasks, they were asked to think out loud. While observing the process, Nielsen's ten heuristics [10] were paid attention to. The testing session was followed by an interview to gain additional knowledge of what is good and what is missing from the overall project, workflow and interface. As a result of the user testing sessions, it was pointed out that users liked how the interface felt, the colour coding of the issues table and that the page was intuitive to use. The most common improvement ideas were as follows:

1. The calendar should start from Monday and follow date formatting of day/month/year.
2. The legend for the table should be opened by default.
3. The count of issues added to the PI should be reflected in the summary tables.
4. Scope changes table field should be renamed to PI scope changes to avoid confusion between sprint and PI scope changes.
5. PI input field should support lowercase letters.
6. Instead of selecting dates for sprints, the users would like just to specify the lengths of sprints by weeks.
7. Users would like to select the team as the first thing.

In addition to the interview, the participants were asked to answer a questionnaire based on System Usability Scale (SUS) by John Brooke [11]. The scale developed by Brook helps to assess the usability of a project while taking into account the context [11]. The scale covers aspects like the need for support, training and complexity while using the product [11]. The users are asked ten questions with five options to answer, ranging from 1 - *Strongly disagree* to 5 - *Strongly agree*. The questions are following:

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think that I would need the support of a technical person to be able to use this system
5. I found the various functions in this system were well-integrated
6. I thought there was too much inconsistency in this system
7. I would imagine that most people would learn to use this system very quickly
8. I found the system very cumbersome to use
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with this system

For calculating the SUS score for each answer, the first items 1,3,5,7 and 9 scale position minus one must be summed. Secondly, for items 2,4,6,8 and 10 five minus scale position must be summed. Thirdly the sum of scores should be multiplied by 2.5 to get the overall SUS score for one participant [11].

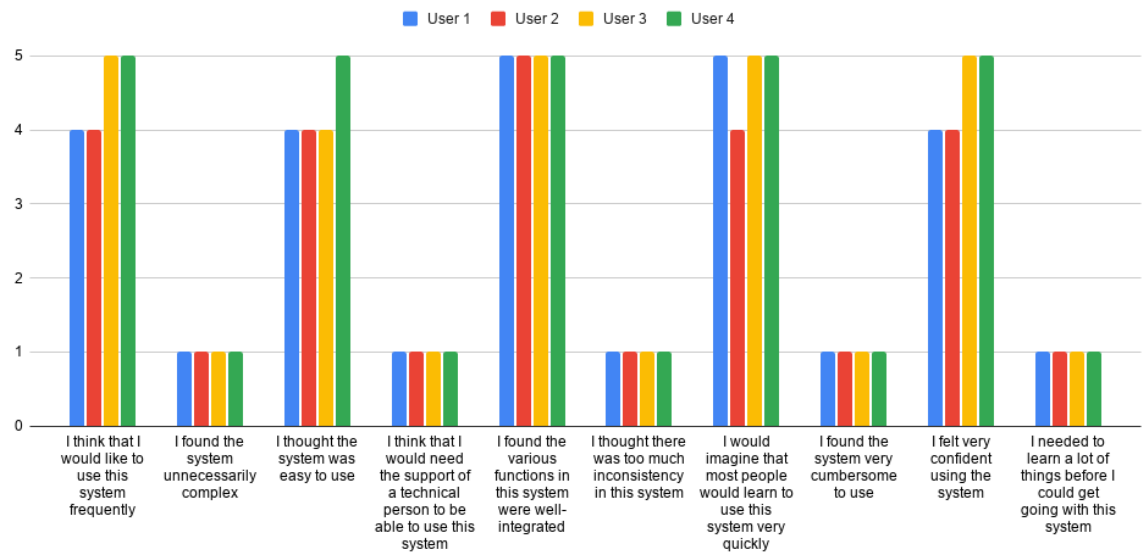


Figure 10. The System Usability Scale questionnaire answers by responders.

For the Planning Accuracy page, the overall SUS score average over all answers was 95 points out of 100, which reflects high satisfaction with the product (figure 10).

5 Solution architecture

The application is separated into different components, helper classes, models and services (figure 11). Each component consists of HTML, CSS and TypeScript files.

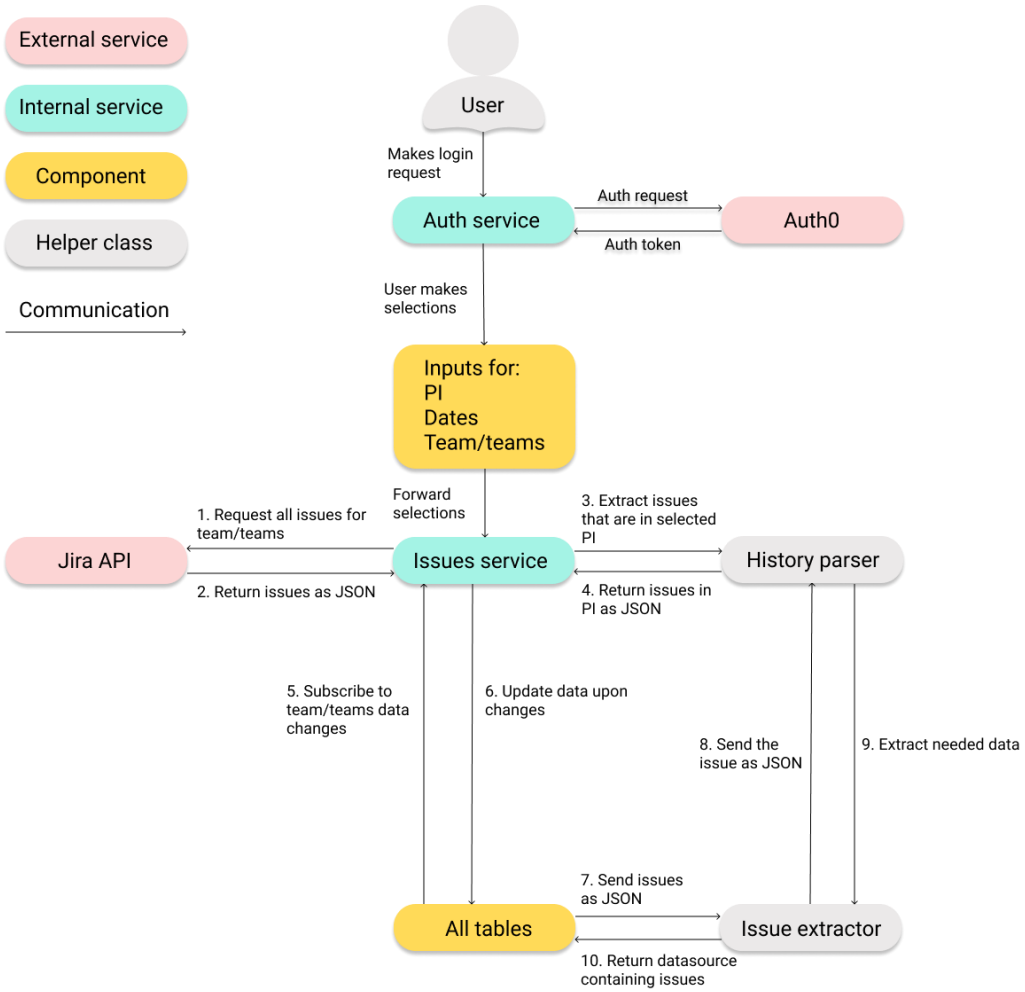


Figure 11. The architecture of the planning accuracy application.

The core of the project is app.component, where it is decided which components to show and in which layout. The following chapter will cover the essential components, functions and how they are tied together.

5.1 Components

The most important components are the ones that make up the tables for the processed data and also the inputs needed to make the API request.

The **sprint-dates** component is used to specify the exact dates for the PI's sprints as that could not be automatically retrieved from the Jira API. There are fields to set the sprint start date for all five sprints and a field to select the PI end date. Usually, the first four sprints are two weeks and the fifth sprint is three weeks. That gave the possibility to fill in all the dates after the user selects the first sprint's start date. After all the other fields are filled automatically, they can still be edited if needed. The suggestion also considers that sprint does not start on weekends

and suggests the start date to be on the following Monday. The dates can only be selected using the calendar to avoid error scenarios with unsuitable inputs.

The **summary-table** component displays the PI data for the selected team. The table uses an Angular Material mat-table[7] component. Figure 5 shows an example of the summary table that has six columns. The first one is labelled *Key* and it contains the IDs of the issues, which are also links to the issues. The second column is *Summary* which is for the short description of the feature. The third column displays the number of the sprint in which the issue was planned to be completed at the PI's start. The next column is called *Revised target* and it is used in case the initially planned delivery sprint is changed to be in some later sprint. The fourth column is labelled as *Status* and it contains the date when the issue was marked as *Ready for UAT,* which is considered to be delivered. It also points out if the issue is rejected. The last column is *Scope changes* which brings out the features added or removed from PI and the corresponding date. The features in the table are also coloured as follows:

- Green – if the issue is ready for UAT before the initial target
- Yellow – if the issue has been rejected
- Light red – if the issue has had a change in *Target sprint* fields
- Red – if the issue has been finished after the end of the initial target sprint or it has not been finished at all
- Blue – if the issue is for offering testing support for some other team

The *DataSource* instance is used to pass the data to the table as recommended by Angular documentation[8]. The class takes care of requesting the data from an internal service called *IssuesService*. It uses a helper class to extract the needed data from the API response and the table's pagination.

The **teams-accuracy-table** table combines the planning accuracy data for all the teams. For each team, it displays the number of total issues in PI, number of issues removed or rejected, number of test support issues, number of issues not delivered by plan, number of issues delivered according to initial plan, and the team's planning accuracy. For calculating the planning accuracy, the number of delivered issues is divided by the number of all issues planned from where the number of rejected, removed and test support tasks is subtracted. The table is sorted by planning accuracies.

5.2 Helper classes

The following classes are used to divide the API response processing into smaller sections and that the methods could be reused between components.

Each issue requested from the Jira API contains a field that contains all the changes done to the issue over time. The ***IssueHistoryParser*** class is used to extract the needed data from the history. First of all, it is used to find from all the issues for the given team the ones that were in the selected PI. This can be difficult as the PI values of tasks can have many changes over time and it must be made sure that they had the correct PI set at the start of the first sprint or they were added during the PI. In addition, it must be made sure that the issue has not been rejected before the start of the PI and it has a Go decision marked in the *GO / NO GO* field. Additionally, the class has methods to extract from issue history data the initial target sprint,

[7] https://material.angular.io/components/table/overview

[8] https://material.angular.io/components/table/overview#datasource

revised target sprint, additions or removals from PI and status at the end of PI. This is the only class dealing with the history data.

IssueColorizer class has a method that takes one *Issue* object as an input and decides what colour it should be coloured. The *Issue* object sets the colours themselves after construction. The *colourise* method returns whether the issue was rejected or removed, test support, had target revised, was completed on time or was not completed on time. The *Issue* object has corresponding colour codes set for these responses. For deciding if an issue was finished on time or not, it takes the date when the status of the issue was set to *Ready For UAT* and checks if that is before the end of the sprint it was planned to be finished in. It also has an option to give some extra days from the end of sprints when issues will still be counted as delivered on time. This is needed as often the Jira is not filled in at the end of the sprint as the day is filled with meetings for many teams. The responsible people will update Jira on the following day or following Monday as usually, the sprints end on Friday. There is no corresponding label in Jira for deciding whether the task is a test support task or not and it is marked as a part of the issue description. There is the following regex pattern for identifying the issues that are test support tasks:

.((t|T)es(t|ting) (s|S)upport|(t|T)es(t|ting)(s|S)upport|(uat|UAT|Uat) (S|s)upport).**

IssueExtractor class takes a JSON list of issues and converts them to *Issue* objects using the aforementioned *IssueHistoryParser*. It also uses the *IssueColorizer* class so that the colour of the *Issue* could be decided.

5.3 Models

Models represent the structure of the data that is used throughout the project. The models can also have methods to process the information once constructed.

Issue model is used to work with issues. It has required fields as *key*, *summary* and *link*. The optional fields are *initialTarget*, *revisedTarget*, *status*, *scopeChanges*, *colour* and *accuracyStatus*. The last one, *accuracyStatus*, is used to choose the colour for the issue and when calculating the planning accuracy. The possible values for the variable and corresponding colours are:

- good – green
- rejectedOrRemoved – yellow
- revisedTarget – light red
- notFinishedOnTime – red
- testSupport – blue

The issues that have revision in the target sprint are marked as lighter red as it is possible that the features were delivered by the end of PI or the target sprint that was set later, but from the planning point of view, they were not delivered according to the initial plan.

The ***TeamAccuracyData*** model contains the PI results summary for one team. Its constructor takes a list of *Issue* objects and will count the total number of issues, number of issues rejected or removed, number of issues that are test support, number of issues that are not delivered according to plan, number of issues that are delivered according to plan and it also calculates the planning accuracy for the team. The planning accuracy is calculated by subtracting the rejected, removed or test support tasks from the total. The number of delivered issues is then divided by the result of the subtraction and multiplied by a hundred.

5.4 Services

There is one service that connects all the components and makes the API requests.

Issues service is the only class that communicates with the Jira API. Other components can subscribe to the *BehaviourSubjects*[9] that contain the response data and update themselves when the response changes. The service has an option to retrieve data for one team or for all the teams together. As the program deals with the history data of the issues, it means that there is a need to query all the existing issues in Jira for every team. These queries can take long. At the time of writing the thesis for the team with most issues, 578, the query took on average 49.3 seconds. The average was counted over five requests for which the time was measured. To improve the situation, the requests to API were made smaller and carried out in parallel. When setting the request limit to 50 issues per request, then the whole data for the same team was queried in 11.3 seconds on average. This meant a 38 seconds improvement for one team and in the case of ten teams, the improvement was even more significant. This also improved the service's scalability as Jira API can return only a certain number of issues with one response[10]. If the user loads some teams' data separately, it will be stored so that when the user requests all teams' summary, the query would not have to be made again for the teams that have already been loaded.

5.5 Authentication

For authentication, the Auth0 framework is used. The implementation is based on Auth0 documentation for single page application[11] and configuration[12]. Auth0 offers a way to direct all your users to their login screen from where, after successful authentication, they are directed back to your application. In the application, there is *AuthService* that is responsible for communicating with Auth0. If it gets the access and ID tokens from the Auth0, then users will be redirected to the main page. It also sets a session for each user, after which the tokens will expire. In addition, it provides a way for users to log out. *Auth-guard* service is responsible for deciding whether or not the users should be able to navigate to the requested page. In case of no authorisation, they will be redirected back to the login screen. Managing the users is done from the Auth0 dashboard[13]. Also, the allowed connections and redirect URLs are managed from the dashboard.

5.6 Used technologies

The technologies used are based on TypeScript and Angular setup. Such a choice was made to increase the developing speed since such setup is well documented and therefore easier to implement. This setup also left open the possibility to integrate the solution to Jira and display it as a subpage there.

---

[9] https://luukgruijs.medium.com/understanding-rxjs-behaviorsubject-replaysubject-and-asyncsubject-8cc061f1cfc0

[10] https://docs.atlassian.com/software/jira/docs/api/REST/7.6.1/#pagination

[11] https://auth0.com/docs/architecture-scenarios/spa-api/spa-implementation-angular2?fbclid=IwAR1eSbRnPWFVJrWCZ3s6d_JXHQnF949317aeIMZcat1Tzpa7pnmzgn4DYo4

[12] https://auth0.com/docs/architecture-scenarios/spa-api/part-2?fbclid=IwAR27KcsymLCGPWKdCaminF1FJiTxOyNM6tEu5BMyOMhSTQ5M9RLSqpLqEfQ#auth0-configuration

[13] https://auth0.com/docs/get-started/dashboard

TypeScript is an open-source language built on JavaScript with the addition of adding static type definitions [12]. Using types assures that an object is of a particular type, resulting in better documentation and validating if code is running as expected [12].

Angular is a component-based development framework built on TypeScript. Angular applications use components and templates as building blocks. Each component contains an HTML template that declares how the component is rendered. Angular supports listening to value changes and events in components and automatically updating the DOM upon changes [13].

Angular Material is a component library that is used in the thesis solution. Angular Material offers well tested and configurable components to use in projects. Also, offering great documentation helped to speed up the development [14]. Some of the components used in the planning accuracy webpage are datepicker, grid, progress bars, toggle, paginator, table and input.

Auth0 is a platform for implementing authorisation. Auth0 offers a sign-in form and registration and can verify users' identity trying to sign in and pass the information back to the application [15]. For granting access and passing the information to the application, Auth0 uses the OAuth 2.0 authorisation framework, a protocol for securely passing users' protected information [16].

Jira REST API is used to retrieve data from the clients Jira Server. It has options to query data for custom fields and labels. Also, it allows setting the range of results requested for pagination. Both basic authorisation and OAuth 1.0 can be used to communicate with the API [17].

5.7 System requirements

For running the planning accuracy application locally, the user should have Node.js[14] and NPM Package Manager[15] installed on his/her computer. The project should be cloned from the repository[16] After cloning, the user should navigate to the project folder and run *npm install*. Once all the installs are made, then running *ng serve* will make the application to be available from http://localhost:4200. By default, the application will not be connected to any Jira API. The configuration can be done in *environments.ts* file. If the API instance requires authentication, then that can be in a *auth/authentication_config.ts* file in the form of *authKey: "Basic auth"*. Depending on the Jira instance, the team names, labels and smaller configurations will have to be made. The Jira should have custom fields *Program Increment* for marking down which PI is the task in, *Target Sprint* which contains the number of the sprint the feature is planned to be finished in and *GO/NO GO* for marking if the feature has the approval to be taken into the PI. Also, the status *Ready for UAT* should be in use.

[14] https://nodejs.org/en/

[15] https://www.npmjs.com/

[16] https://github.com/vanatoomas/PI-planning-accuracy

6 Conclusion

The aim of the thesis was to make assessing the planning accuracy easier when using Scaled Agile Framework methodology and Jira issue management system. To achieve that, the author of the thesis held a series of meetings with a large corporate client who is using such a framework. During the meetings, it was clarified what analysis must be done on Jira data and the calculations needed to get the planning accuracy for development teams. On planning, each team must estimate each feature with the sprint in which they are planning to deliver the feature. For calculating planning accuracy, the features that are completed by the end of the initially planned sprint are divided with all the features in Program Increment.

The solution developed within the thesis is a webpage that requests the needed data from Jira, extracts the issues that were in Program Increment, finds if the feature was delivered by initial plan and visualises the results. The page lets users to see on feature level how each team performed in the Program Increment by displaying a list of planned features, when were they estimated to be finished, whether the feature was delivered according to the plan and the planning accuracy of the team. It also gives the user the possibility to retrieve the planning accuracies for all the teams at once and compare them in a table.

For evaluating the Planning Accuracy page, four individual user testing sessions were carried out with interested parties. The users were asked to conduct series of tasks that were observed. The session also included an interview with the user. The focus of the observation and interview was to find if the application was understandable and intuitive to use. Although the participants were satisfied with the page, they pointed out that the calendar modal could be more user-friendly if the calendar would start from Monday, follow format day/month/year and instead of dates, the duration of sprints could be used. In addition, participants were asked to fill in a survey for evaluating the usability. The application scored 95 points out of 100, which reflects the satisfaction of the participants.

As for future work, there are few additions that could be made to the application but did not fit the thesis scope. The first one would be to keep the history of all the planning accuracies that would give the possibility to monitor teams progress over time. The second improvement would be to add an export option for extracting the data related to planning accuracies. Also, sorting options could be added to the issues table that is displayed for one selected team.

Currently, the program is configured to work with one specific Agile Release Trains Jira. The Jiras of other trains could be examined to make the configuration of the application easier for new customers.

References

[1] Digital.ai 2020. 14th Annual State of Agile Report https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494 (09.12.2020).

[2] Leffingwell, Dean 2020. Welcome to Scaled Agile Framework 5.0 https://www.scaledagileframework.com/about/ (09.12.2020).

[3] SAFe methodology. https://www.scaledagileframework.com/program-increment/ (09.12.2020).

[4] Scaled Agile SAFe whitepaper. https://www.scaledagile.com/resources/safe-whitepaper/ (15.04.2021)

[5] Release Train Engineer responsibilities. https://www.scaledagileframework.com/release-train-engineer-and-solution-train-engineer/ (06.03.2021)

[6] Leffingwell, Dean 2011. Agile Software Requirements: lean requirements practices for teams, programs, and the enterprise, 309-313

[7] Ticket in the Atlassian system. https://jira.atlassian.com/browse/JRASERVER-27641 (09.12.2020).

[8] Issue History plugin. https://marketplace.atlassian.com/apps/1220385/issue-history?hosting=cloud&tab=overview&fbclid=IwAR1eBMXP8-LI_wL_AqmSmnrv2XtxyCjWSXogyJkL5qnv9L_nqs_52QIz4aA (06.03.2021)

[9] Issue History Timeline plugin. https://marketplace.atlassian.com/apps/1220310/issue-history-timeline?hosting=server&tab=overview (06.03.2021)

[10] Nielsen, J. (1994b). Enhancing the explanatory power of usability heuristics, 152-153.

[11] Brooke, John 1996. Sus: a quick and dirty usability scale. Usability evaluation in industry, 189

[12] TypeScript https://www.typescriptlang.org/ (06.03.2021)

[13] Angular documentation, https://angular.io/guide/what-is-angular#what-is-angular (06.03.2021)

[14] Angular Material, https://material.angular.io/ (06.03.2021)

[15] Auth0, https://auth0.com/docs/get-started (09.03.2021)

[16] OAuth 2.0, https://auth0.com/docs/protocols/protocol-oauth2 (09.03.2021)

[17] Jira REST API, https://developer.atlassian.com/server/jira/platform/rest-apis/ (09.03.2021)

Appendix

I. User testing procedure

Study goals

1. Identify the aspects that could be improved for the workflow
    a. Answers the question: How should the workflow be designed to make finding planning accuracy more convenient?
2. Identify the shortcomings of UI
    a. Answers the question: How could the layout, colours and explanations in the application make more understandable and easier to comprehend?

Materials

Description of tasks as pdf, Microsoft Teams, consent form as pdf, post-survey, access to the Planning Accuracy program running in VPN protected server, user-created for the participant to log in to the application

Introduction

As Product Owner/Agile Release Train Engineer/Analyst, you have been selected to take part in the study. This study is conducted by Karl Toomas Vana, who is a third-year bachelors' student at the University of Tartu.
The aim of the study is to evaluate how to improve the usability of the Planning Accuracy page. The purpose of the page is to make the comparison between teams plans and the actual outcome of the Program Increment simpler.
The focus of the study is to find shortcomings in the workflow of searching planning accuracy. Also, to identify whether there are any improvements needed in the UI for a better understanding of the program.
In the study, you will be asked to complete a series of tasks that are sent to you beforehand. The tasks focus on the main functionality of the application, that is finding the planning accuracy for any given team along with detailed information about the features involved and also making the comparison of planning accuracies between all teams. While completing the tasks, you will be asked to share your screen.
After going through all the tasks, we will conduct a short interview to analyse the outcomes of the session. The session will be recorded and results used to improve the given application.

Consent form

See appendix II.

Test procedure

1. Introduce the study
2. Confirm participant has received the tasks descriptions
3. Confirm the participant has access to the Planning Accuracy page

4. Introduce the application
5. Explain the upcoming tasks and their order
6. Ask the participant for consent to record the session
7. Start recording as ask the participant to carry out the tasks in the given order

Observation

I will observe the entire process and take notes. The focus of the observation is to notice if the participant has any problems using the tool. In addition, I will look if the user follows the workflow we would expect, if there are any problems how will the participant overcome them and if there are some of the tasks that take longer than expected. Also, Nielsen's ten usability heuristics[17] should be paid attention to:

1. Visibility of the system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognise, diagnose and recover from errors
10. Help and documentation

Post-interview

1. Ask about the overall feel of the project
    a. Point out one thing that the participant liked and disliked about the page.
    b. Was there anything that created confusion? If something was noticed on observation, then address that topic directly
2. Ask about the workflow
    a. Was there anything that would have made the flow more convenient?
    b. Did the user feel stuck at some step?
3. Ask about the interface
    a. How did the participant like the placement of the elements?
    b. Was there anything distracting or off?
4. Ask about improvements
    a. Is there anything the participant would like to add to the page that he/she would expect from the planning accuracy page but was missing?
    b. Is there anything else the participant would improve that has not been mentioned earlier?

[17] Nielsen, J. (1994b). Enhancing the explanatory power of usability heuristics, 152-153.

Post survey

The survey will be based on the System Usability Scale by John Brooke[18]. It will contain the following questions that can be rated on a scale from 1-5, where 1 means strongly disagree and 5 strongly agree:

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think that I would need the support of a technical person to be able to use this system
5. I found the various functions in this system were well-integrated
6. I thought there was too much inconsistency in this system
7. I would imagine that most people would learn to use this system very quickly
8. I found the system very cumbersome to use
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with this system

Overall procedure

1. Send the tasks descriptions, consent form and link to the application beforehand so the participant could verify that they have access to their account
2. Introduce the session procedure and ask for consent
3. Once the participant has given consent, start recording
4. Introduce the aims of the study and procedure in more detail
5. Introduce the Planning Accuracy page
6. Verify that the participant has received the task list
7. Explain the tasks and order
8. Ask if the participant has any questions
9. Ask the participant to share screen
10. Ask the participant to start from the first exercise
    a. Remind the participant to think out loud
    b. Offer to provide help if the participant should get stuck and cannot find the solution themselves in a reasonable time
11. After all tasks are completed, carry out post-interview
12. Ask the participant to fill in the post-survey
13. Thank the participant for the contribution
14. Stop the recording
15. Save the recording file and notes made during the session

The tasks

Imagine it is the end of PI-1 2021 and you are analysing the results. First, you would like to check how particular teams did in the previous sprint (PI-4 2020) and later compare it to the latest PI results. Following are the tasks that you need to complete to find the info that you are interested in.

1. Find the planning accuracy for team Selleri in PI-4 2020. The dates for the PI were:

[18] Brooke, John 1996. Sus: a quick and dirty usability scale. Usability evaluation in industry, 189

a. Sprint 1 start 14.09
b. Sprint 2 start 28.09
c. Sprint 3 start 12.10
d. Sprint 4 start 26.10
e. Sprint 5 start 09.11
f. Sprint 5 end 26.11

2. Find how many features did team Geneva not deliver in PI-4 2020
3. Find how many tasks were added to PI and how many rejected for team Self Care
4. Find if the team with the best planning accuracy had any tasks that were removed from the PI. If yes, then find the ID's of the corresponding JIRA issues
5. Find how the best team of PI-4 2020 did in PI-1 2021. The dates for PI were:
   a. Sprint 1 start 14.12
   b. Sprint 2 start 04.01
   c. Sprint 3 start 18.01
   d. Sprint 4 start 01.02
   e. Sprint 5 start 15.02
   f. Sprint 5 end 05.03

II. Consent form

Consent to Act as a Participant in a Study

Study title: Validating the usability of Planning Accuracy page

Investigator: Karl Toomas Vana

Introduction: As Product Owner/Agile Release Train Engineer you have been selected to take part in the study. This study is conducted by Karl Toomas Vana who is a third year bachelors' student at the University of Tartu.
The aim of the study is to evaluate how to improve the usability of the Planning Accuracy page. The purpose of the page is to make comparison between teams plans and actual outcome of the Program Increment simpler.
The focus of the study is to find shortcomings in the workflow of searching planning accuracy. Also to identify whether there are any improvements needed in the UI for better understanding of the program.
In the study you will be asked to complete a series of tasks that are sent to you beforehand. The tasks focus on the main functionality of the application, that is finding the planning accuracy for any given team along with detailed information about the features involved and also making the comparison of planning accuracies between all teams. While completing the tasks you will be asked to share your screen.
After going through all the tasks we will conduct a short interview to analyze the outcomes of the session. The session will be recorded and results used to improve the given application.

Participation requirements: Product Owners or Release Train Engineers who are interested in the planning accuracy of the Program Increment

The expected duration of the study: The study will take about 30-35 minutes of your time.

Risks and Benefits: There are no risks to the study but application made based on the participants' feedback could be of value to the interviewee.

Privacy and Confidentiality: In order to protect the participants' identities during this study the investigators will follow the following procedure: The original recordings will only be accessible to the investigator. Audio contained in the recordings will be transcribed, potential identifiers will be removed or aggregated and the original recordings will be deleted afterwards.
Your data and consent form will be kept separate. Your consent form will be stored securely and will not be disclosed to third parties.
By participating, you understand and agree that the data and information gathered during this study may be used by the participating university for publication purposes. However, any identifiable information will not be mentioned in any such publication or dissemination of the study data and/or results.

Questions about the Study: If you have any questions, comments, or concerns about the study either before, during, or after participation, please contact Karl Toomas Vana karltoomas.vana@proekspert.ee.
Voluntary Participation: Your participation in this study is voluntary. You may discontinue participation at any time during the study.

I am of age 18 or older. I have read and understood the information above and I want to participate in this study:
☐ Yes ☐ No

Participant: The above information has been explained to me and all of my current questions have been answered. I understand that I am encouraged to ask questions, voice concerns or complaints about any aspect of this study during its course, and that such future questions, concerns or complaints will be answered by a qualified individual or by the investigator listed on the first page of this consent document.

Investigator: I certify that I have explained the nature and purpose of this study to the participant, and I have discussed the potential benefits and possible risks of study participation. Any questions the participant had about this study have been answered, and we will always be available to address future questions, concerns or complaints as they arise. I further certify that no component of this study has started before this consent form was signed.
Participant_____
Investigator_____

III. License

Non-exclusive licence to reproduce thesis and make thesis public

I, Karl Toomas Vana,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, **Visualising Jira data for planning accuracy**

supervised by Alexander Nolte and Ezequiel Scott

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Karl Toomas Vana

27.04.2021