Design and Evaluation

- Streamlining the handling of R&D projects

Lolita Olesen

2016

Master's Thesis

Department of Design Sciences
Lund University

Abstract

Agile software development promotes process adaptability throughout the life cycle of a project. A project is often broken down into smaller tasks and each task involves a cross-functional team working with all functions. There are many different agile methods, which focus on different aspects of the software development life cycle. Two of these agile methods are Kanban and Scrum, which focus on managing software projects. The basic unit of development with Scrum are the sprints, which are a specific duration. Each sprint demands planning tasks to be solved beforehand and an estimated commitment of tasks for the sprint goal. To help the team to estimate how much work they can do in a future sprint, a collection of historical velocity data is needed. This is often done by drawing the sprints in a chart to visualize the data. To help the teams to stay on track during each sprint, other charts can provide visibility of their on-going process. The key for success in Kanban is to visualize the workflow to improve the process. A Kanban board is often used in combination with different types of charts and tables.

The aim of this Master thesis was to explore a combination of different charts and tables to increase the efficiency in R&D projects for agile development. This was done by creating and evaluating a prototype and then measuring the usability of it. The first prototype was developed and then tested on the actual target group at a company. The participants evaluated the tool and suggested the further development of the prototype. The final prototype was then developed based on the first one. A usability test was conducted on a potential target group, which haven't used similar tools before. The results from the prototype were then evaluated. In both the first and the second tests, usability and efficiency were in focus. The conclusion of the final usability test was that there was a high level of usability although the tool was custom made for projects using Kanban and Scrum. The tool also demanded some basic level of knowledge to make it usable.

Acknowledgement

I thank Joakim Eriksson, my supervisor at LTH, for his support. My supervisors at Axis, Henrik Andersson and Fabrice Coulon for their feedback and trust. The Tools team on Axis for all their support and guidance, especially Bekim Berisha, Stefan Gangefors and Alexander Lapajne for their patience and teaching me everything I know about developing in Python. Robert Svensson for helping me to stay on track and other employees at Axis who has helped me with great ideas. Finally, Axis Communications for letting me do this master thesis project.

Contents

Glossary

Efficiency - Once the users have learned the design, how quickly they can perform the given tasks. (Norman, 1990)
Errors - How many errors the users make, how severe are these errors, and how easily they can recover from the errors. (Norman, 1990)
Hi-fi - high fidelity
Interaction design - Support the way people communicate and interact in their everyday by designing interactive products. (Norman, 1990)
Lo-fi - low fidelity.
Learnability - How easy is it for users to accomplish a given task the first time they encounter the design. (Norman, 1990)
Memorability - When users return to the design after a period of not using it, how easily they can re-establish proficiency. (Norman, 1990)
Satisfaction - How pleasant the design is to use. (Norman, 1990)

1 Introduction

1.1 Background

This master thesis was developed at Axis Communication in Lund, Sweden. Axis is a Swedish manufacturer of network cameras for use in the field of physical security and video surveillance. The team "Tools" is in the R&D sector at Axis and works with agile software development. They create and handle tools for internal usage. Departments at Axis Communication use different tools for handling internal issues. They use different approaches for solving the issues depending on the department, team and project. Most projects use some sort of chart or table depending on which type of agile software development that is applied. The charts and tables are used for several purposes, to visualize the workflow, to make estimations or to collaborate within the team of the project.

What they all have in common is the need for a tool that visualizes the efficiency of their handling of issues. There is currently no tool for knowing how correct the estimations are or what the average time to solve issues is. Estimations are based on experience from individuals and not the actual data. This makes it harder to know where the bottle necks are and where improvements are needed to increase the efficiency.

Data for how the issues are handled is available in different tools, but it is too time consuming to interpret the data manually. Therefore, there was a need for a tool to help the teams to interpret the data by creating charts and tables to increase their efficiency.

1.2 Research question

Can the efficiency of R&D projects increase by creating a tool that visualizes the data of the handling of issues?

1.3 Purpose, goal and scope

The goal of this master thesis was to increase the efficiency of handling the issues in agile software development. This was done by creating a website for visualizing the streamlining of the handling the issues. The thesis conducted a software design process with initial research, prototyping, implementation and testing. The purpose of testing was to measure the usability of the tool. Since the tool was supposed to be used by several operators, using different approaches and interested in different information, the challenge was to find a solution, which was suitable for several situations. To complete the task the thesis was divided into several phases, which was estimated early to meet the deadline of 20 weeks.

1.4 Approach

1.4.1 Literature study

A thorough literature study was done at the beginning of the research to create a broad foundation for this master thesis. The research was oriented at agile software development, different approaches of it and how to make it efficient. Three scientific articles were used as a basis for working with agile development and how to make the process of it more efficient. These are presented in section 1.4.1.1. As the development problems arose, the design choices were based on literature studies on design, usability and interaction.

1.4.1.1 Related work

Hollerup, Hermansson and Green wrote a thesis in May of 2015 where they identified and compared different methods in agile software development. Their thesis presented the challenges of working agile and addressed the improvements of the livelihood projects. Research questions, which this paper is based on, "What is the biggest challenge for agile software projects?". The conclusion was that the greatest challenge for agile software projects was the estimation of projects directly or indirectly (Hollerup, Hermansson and Green, 2015).

Another scientific article that was a basic for this master thesis was Berndt and Jönssons who investigated how projects' usefulness is connected to the agile software development process. The focus was on Scrum, which is a agile development method. They presented how Scrum can provide project usefulness. The different features in a Scrum project process that results in the most project usefulness are working with sprints and the daily Scrum (Berndt and Jönsson, 2011).

Jönsson investigate how agile development and user-centered design can be combined in a development process. The conclusion of the master thesis was that the development process needs to combine user-centered design by more involvement of usability evaluation methods and end-users at an early stage (Jönsson, 2013).

1.4.2 Design approach

This master thesis is divided into five parts, which is described in figure 1.4.2.

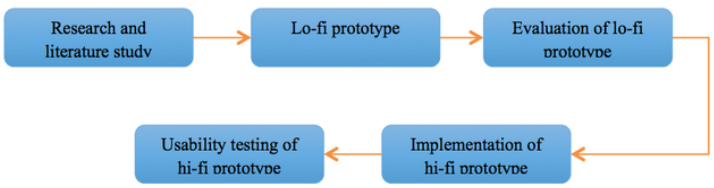

Figure 1.4.2 displays the design approach of this master thesis.

The first phase of the master thesis was about gathering as much information of the current tools that was available, what data that was going to be processed

and what research has been done before. Based on that, a lo-fi prototype was created. There two phases was an iterative process. The lo-fi prototype was designed on papers.

The next phase was the third one, evaluation of the lo-fi prototype. This was done on current employees at Axis that uses similar tools on projects. This evaluation was then the basis for the fourth phase, implementation of the hi-fi prototype. All of the implementation and the design of it was an iterative process. When the implementation was finally done, the final phase begun, which was usability testing of the final hi-fi prototype. The test results were evaluated and then summarized.

2 Theoretical background

2.1 Agile development

Agile software development is a set of principles that are used within software development in which requirements and solutions evolve through collaboration between self-organizing and cross-functional teams (Agile Alliance, 2013). In R&D at Axis, agile software development is a frequently used method for solving problems and issues. Two of the most popular agile methods are Kanban and Scrum.

2.1.1 Kanban

What is Kanban?
Kanban is a Japanese word for "visual card" and is a lean approach to agile software development. It's based on a very simple idea that work, which is in process, should be restricted until existing work is delivered or is blocking the process. This visualizes the issues and problems and makes everyone involved to focus on handling the problems to restore the flow. Kanban uses a visual control mechanism to follow work flowing through the various stages in the value stream. This is normally approached by a simple board or with named columns where you can attach different stickers, which describe the different tasks. Figure 2.1.1 of a typically Kanban board is shown below (Anderson and Reinertsen, 2004).

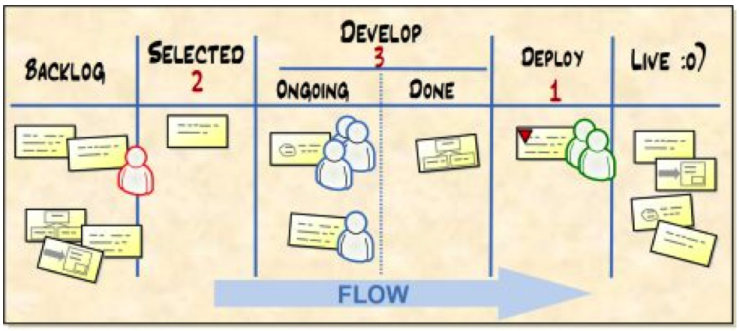

Figure 2.1.1 shows a typical Kanban board (Anderson and Reinertsen, 2004).

How does Kanban work?

1. Visualize the workflow.
   a. Split the work into pieces, write each item on a card and put on the wall.
   b. Use named columns to illustrate where each item is in the workflow.
2. Limit WIP (work in progress) – assign explicit limits to how many items may be in progress at each workflow state.
3. Measure the lead time (average time to complete one item, sometimes called "cycle time") optimizes the process to make lead time as small and predictable as possible.

(Anderson and Reinertsen, 2004).

What are the benefits of using Kanban?

- It exposes problems in real-time. Kanban visualizes bottlenecks, queues, variability and waste, all of which influence the organization's performance in terms of both quantity of work and time to deliver it.
- It provides team members and external stakeholders an insight into the impact of their actions. This leads people to collaborate instead of just handling their own part and encourages discussions of improvements.
- The effect of restricting on-going work is that the cycle times become more predictable and accurate delivery. By providing a way to do agile software development without having to use time-boxed fixed-commitment iterations, Kanban is useful for situations where limited time-box don't make much sense. Since there is a limit on the number of tasks instead of the time on solving it, bottlenecks can be avoided and time saved anyway.
- Kanban encourages prioritization of new work and delivery of work performed. Typically the teams hold short and frequent meetings where it's agreed on a reasonable rhythm during meetings with stakeholders when it is determined what to do next. This leads to a simple question usually getting a speedy and high quality response. Demand nature means that the work commitment is delayed until the last moment to decide. This increases mobility by managing expectations, short cycle times from commitment to delivery and eliminating rework because the risk that priorities change is minimized (Kniberg and Skarin, 2013).

2.1.2 Scrum

What is Scrum?
Like Kanban, Scrum is an agile method that optimize the development. The main focus should be on what is supposed to be developed rather than how it is going to be done. What characteristics Scrum is the short iterations (sprints) where every task is assigned with a priority to a person or a team (Kniberg and Skarin, 2013).

How does Scrum work?

1. Create a backlog. All tasks for the project are summarized in a backlog, which is the same as a "to-do-list". All tasks are divided into hand able pieces and are then prioritized. The product owner normally does this.
2. Create a Sprint backlog. The tasks with the highest priority are then put into a Sprint backlog. A sprint is normally finished within two to four weeks. Therefore, it's important to estimate how many task the team can deliver within the sprint. If there are more tasks in the sprint backlog after the sprint is done, the task is moved to the next sprint.
3. Daily Scrum.
   Every day, the team has a short meeting to answer the following questions:
   - What have you done since the last meeting?
   - What will you do until the next meeting?
   - Is there anything preventing you?

What are the benefits of using Scrum?

- Focus on customer benefit and customer communication. It normally gives the customer more control over status of the project. The customer can easily understand and explain exactly what they want by prioritizing the tasks.
- Transparency - everyone sees the work and priorities set. Like in Kanban, this may lead team member to collaborate instead of just handling their own part and encourages discussions of improvements. Because the team member is participating in prioritizing the tasks, it's more likely to get an accurate prediction.
- Short iterations of two to four weeks. This makes it easier to estimate how much time the task will consume. However, in case of unpredictable events, this may be a problem.
- Clear division of responsibilities between the development team and the rest of the organization. Since the team member has more responsibility than normally, it encourage the members to get involved in the project. There is still division of responsibility so that all team members still can focus on what they can control (Kniberg and Skarin, 2013).

2.1.3 Kanban vs. Scrum

Both Scrum and Kanban are very adaptive. However, Scrum is more prescriptive than Kanban for several reasons:

1. Since Scrum is time-limited to the sprints and gives thus less room for alternatives. Kanban doesn't use time-limitation, just estimation that is done in its own pace.
2. Scrum prescribed three roles within the project:
    - The product owner who set the product vision and decides the priorities.
    - The team who implements the product.
    - The Scrum master who provides process leadership within the project.

In Kanban, there is no such thing as set roles. That, however, does not mean that you can't have fixed roles, it only advocates that you don't have to.

1. Scrum is limiting on-going work per iterations while Kanban is limiting their on-going work per status. In Kanban, each state cannot contain more than a set of tasks per status. There is no such advocated limit in Scrum. Although in most cases, there is an invisible limitation for not creating an overwhelming feeling to have too much running simultaneously. When using Scrum, the on-going work is limited per iteration. The number of tasks per sprint is hard to predict in the beginning of the projects, but is normally later easier to estimate based on historical sprints within the project (Berndt and Jönsson, 2011).
2. Scrum resists changes within an iteration while Kanban is more flexible. If a task in the backlog or a new task is presented with a high priority, it still can be executed in Kanban even though other task is on-going. Since there is a limitation on how many tasks that can be in the same status, another task can be put aside and then proceeded when there is free space in the state again. In Scrum it's harder. Since the sprints are short, the new task will most likely be preceded in the next planned sprint.
3. Larger tasks don't always have to be broken down into smaller piece in Kanban. In Scrum, all tasks have to be done within the sprint. Therefore, if there is on task that won't be finished within the sprint, it has to be broken down to smaller tasks. In Kanban, there are no rules that decides the size of the task. It can sometimes be more efficient to finish the whole task as a stream instead of taking small pieces of it (Hollerup, Hermansson and Green, 2015).

2.2 Iterative development

When developing iteratively, the iterative life cycle model does not attempt to start the development with a full specification of requirements. It begins by specifying and implementing a part of the system, then review it and then identify further requirements, which is brought to the next iteration. This process

is then often repeated in small cycles, each cycle represented by a new version until the system is completed. Iterative development is quite useful when major requirements are defined but smaller details can evolve with time. By improving a product iteratively defects can be tracked at early stages. When using prototypes, iterative development is preferred (Cederholm, 2006).

2.2.1 Low-fi prototyping

A lo-fi prototype is a representation of what could be implemented. It's a prototype that has a few or no working parts at all. A lo-fi prototype is created very early in the design process. The advantages of using a prototype are plenty and are often a crucial part for creating a new product. By drawing the design on a piece of paper, the developer can determine usefulness and feasibility and get a quick feeling of the product without having spent too much time. By doing tests on a lo-fi prototype, the developer can get feedback early in the design process, and base essential design decisions based on the results. The test participants are more likely to feel freely to criticize a lo-fi prototype than a finished product since the prototype is a more distant product than a real one. When presenting a low-fi prototype, one must only show the basic functionality for the test person. The reason why is because the test of the prototype is a perfect way to know how to implement the functionality depending on how the test person reacts on the test. During the test the test person could e.g. press a button. When pressing the button that still isn't implemented with any function the test leader can ask the test person what the test person expected to happen when pressing the button. That way the developer knows what the button invites the user to do. This method is referred to as "horizontal representation" (Rubin and Chisnell, 2008).

Some important aspects for testing with a lo-fi prototype:

- Test the prototype with the same method as testing on a finished product.
- Choose test participants who have the same characteristics as the target users of the product.
- Draw all buttons, menus and features on the lo-fi prototype to make it as close to the supposed product.
- Since the features don't work as a real product, the test leader must stage-manage the features by e.g. show the supposed event that would occur if the test person presses a button.

Using lo-fi prototyping for deciding design decisions and using it for early tests basically efficient and cheap investment with a low risk to experimental (Rubin and Chisnell, 2008).

The design is

2.2.2 Hi-fi prototyping

A hi-fi prototype is a prototype that is closer to the real product but still easily disposable. Instead of using drawings on a piece of paper, some features or

interactions should be implemented. Some important aspects for testing with a hi-fi prototype:

- Even though all parts aren't implemented, it's better if the one that isn't finished still look properly live. For e.g. a button could be visible but nothing happens if the user presses the button.
- It's often harder to handle the test session than testing a lo-fi prototype or even a finished product since there are many situations where the test person is expected to test features that isn't implemented yet. This is something that the test leader has to explain every time the event occurs and can make the test person feel a bit annoyed.

Basically, hi-fi prototyping can be harder test on since lo-fi prototype is easier to make the test person to be honest and a finished product conduct at features and does not need any further explanations (Rubin and Chisnell, 2008). The hi-fi prototype of this master thesis will become the final product for this master thesis.

2.3 User-centered design

The concept of user-centered design is more of a philosophy, which is founded on making the product usable and understandable by emphasis on the user's needs and interests. The user should intuitive figure out what the system can do and how it can be done (Norman, 2008).

User-centered design should make it easy for the user to determine what actions are possible. This is done by making the conceptual model of the system visible, also alternative actions and the results of the actions. It should also be easy to follow natural mapping (Norman, 2008). Natural mapping is presented in the paragraph "Natural mapping".

According to Jönsson, the main similarity between the User-centered design and agile software development is their common aspect of developing high quality software for the end customer. However, a difference between them is the focus on customer respective end-user. In agile development the customer the real customer for the project or a representative for the customer. Agile development also values collaboration and communication between the customer and development team. In User-centered design, the end-users are the persons who are using the system. There need to be a distinguish between the customers and the end-users when integrating agile development and User-centered design. For e.g. in agile software development the focus is on satisfying the customers need. Those needs are often changed during the developing process. This focus can lead to problems with user experience if the customer is not the real end-user. This distinguishes between customers and end-users are one of the cornerstones in successfully integrating User-centered design in an agile context (Jönsson, 2013).

2.3.1 Natural mapping

The word “mapping” in technical terms means the relationship between two things. Mapping should be easy to learn and always remembered. An example for mapping is to control something, which results in a movement, like steering a steering wheel clockwise, and the car turns to the right. The technical term “natural mapping” takes advantages of physical analogies and cultural standards for immediate understanding. A natural mapping could also be theme or concept, like creating a program's features all the same. In that case, the user only has to learn how to use the features one time, and the second will come as natural (Norman, 1990).

2.3.2 Three conceptual model

The three conceptual model is a concept that is appropriate for the user, which captures the most important operations when interacting with a product. It can be described as a mental model with three distinguished models: the design model, the user's model and the system image, see figure 2.3.2. The design model is the conceptualization of the designer describing the system. The second model, the user's model, is the user's conceptualization of the system. The system image is the result of the combination of the design model and the users' model. Figure 2.3.2 displays “Three aspects of mental models”.

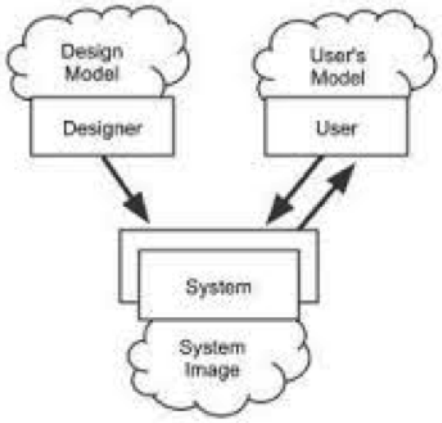

Figure 2.3.2 displays Donald Normans ”Three conceptual model (Norman, 1990).

2.3.3 Mental models

Donald Norman described that mental models are what people really have in their heads and guides their use of things. He also describes that the mental model of a system is based on interpreting the systems perceived actions and its visible structure. This means that if a system is incoherent then the user will not be able to use the system easily. This can then result in the system becoming contradictory and perceived as less usable. (Noman, 1990)

2.4 Feedback

Feedback is to send back information to the user about what actions that has been done. The feedback itself helps the user to confirm what actions that have been made. It is just as important to send feedback when the user does something wrong as if the user does something right. For example, if the user click on the wrong button and doesn't receive any response, then the user would probable press it again since the reaction is normally that the user didn't press correct. Also, when the user presses the right button and doesn't receive any feedback, then even if the user knows that they did it correctly they would most likely press it again out of frustration. Without any feedback, the user experiences frustration over a product or system, even though there is nothing wrong with it (Norman, 1990).

2.5 Data visualization

Here is computer-generated graphics of complex data in focus that are interactive and dynamic. Then handling data visualization, it is important to amplify human cognition and the goal is the help the users to see patterns for making the system more useful. When creating a design of the webpage, navigation, speed, legibility and overall ease of using the webpage are keywords for making great product (Rogers, Sharp and Preece, 2011).

2.5.1 Different types of charts

2.5.1.1 Burndown chart

To visualize the status of the sprint, normally a Burndown chart is used. The start value is set on the number of task in the sprint backlog. Then a linear line is drawn down to the finish date of the sprint. Figure 2.5.1.1 displays a typical Burndown chart.

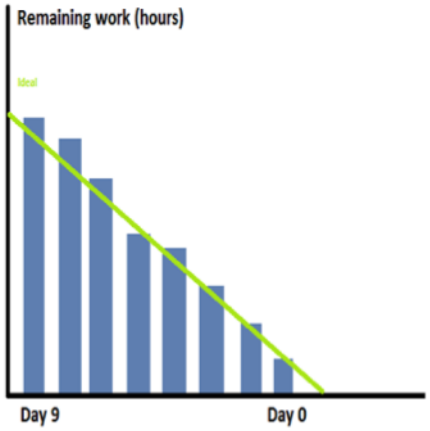

Figure 2.5.1.1 displays a Burndown chart.

2.5.1.2 Velocity chart

A Velocity is another tool that is frequently used in Scrum. It displays sprints in pair, one sprint for the committed points and one for the actual completed points per sprint. Figure 2.5.1.2 displays a typical Velocity chart.

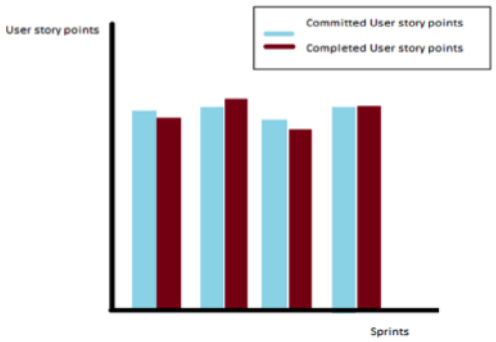

Figure 2.5.1.2 displays a Velocity chart.

2.5.1.3 Pie chart.

A pie chart can display the percentage of how the different categories are distributed. The categories can be divided based on time or the number of tasks. Figure 2.5.1.3 displays a typical pie chart.

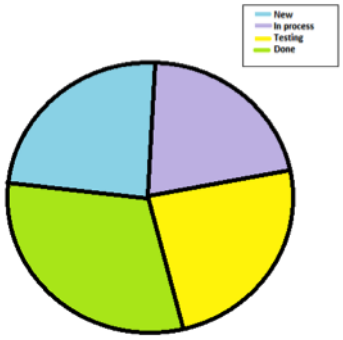

Figure 2.5.1.3 displays a Pie chart.

2.5.1.4 Throughput chart

A throughput chart is a chart that displays all status on a time interval. The y-axis shows the number of items and the x-axis shows the time. The next figure, 2.5.1.4, displays a typical throughput chart.

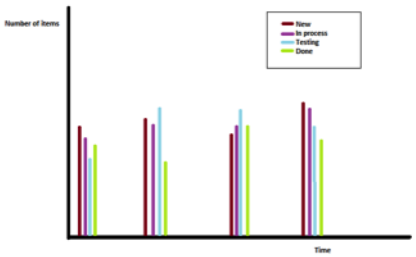

Figure 2.5.1.4 displays a Throughput chart.

2.5.1.5 Cumulative chart.

Cumulative means how much so far. The chart displays how much that is done in each category over time. Figure 2.5.1.5 displays a typical cumulative chart.

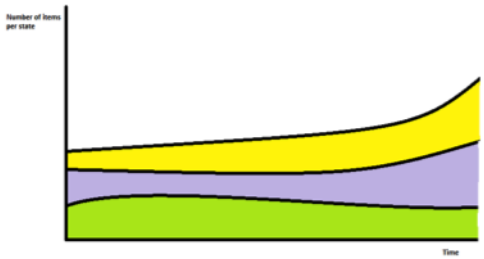

Figure 2.5.1.1 displays a Cumulative chart.

2.6 Usability

Usability can be evaluated on any artifact a human can interact with. From the user's perspective, usability is an important aspect for enjoying the product. However, usability is often only a problem when a product is lacking or absent of it. When something contains a high usability, the usability is most commonly not mentioned (Rubin and Chisnell, 2008) A good usability suggests efficiency, effectiveness, safe to use, easy to learn, easy to remember and have good utility (Rogers, Sharp and Preece, 2011).

2.6.1 Why measure usability?

The main purpose of measuring usability is to find out how participants interact with a system or a product. The findings of the measurement can then help the developer to make crucial decisions (Rogers, Sharp and Preece, 2011).

2.6.2 Usability testing

The term usability testing is often used in user-centred interaction design for evaluating a product or system by testing it on users. The test persons are a representation of the target audience for evaluates the degree of usability the product possesses. The range of usability tests is truly considerable, from basic 5 minutes test to deep and complex tests that can take up to days to execute. Each test approach has different objectivities (Rubin and Chisnell, 2008)

2.6.3 System Usability Scale

SUS (System Usability Scale) is a valuable tool to measure usability of systems. It is divided into 10 statements where every statement has 5 response options, which is graded from "strongly disagree" to "strongly agree". The questionnaire is simple and provides a global view of subjective assessments of usability. It is often used after a user has used the system but it's important that the questionnaire is finished before any debriefing or discussion with the participant. All the odd-numbered questions are formulated with a positive undertone like "I think I would like to use this system frequently in the future". All the other questions are formulated with a negative undertone like "I found this system unnecessarily complex". All questions are then ranged from 0-4. To summarize the score of the questionnaire, all odd-numbered questions are calculated with the score contribution minus 1 point. All the even-numbered questions are calculated with 5 minus the score contribution. After summarizing the points, the sum is multiplied with 2.5. The SUS score has a range from 0 to 100, and a score over 68 point is considered as good usability (Brooke, 1986).

2.6.4 Initial usability testing

An initial usability test comes in handy for finding errors and preventing them before the actual usability test. Performing the test on a pre-test person, also known as a pilot test, can do it. There are several advantages of conducting a pilot test is several:

- Errors can be prevented.

- If any instructions are vaguely described, the test pilot can help to make it easier to understand.
- An opportunity to test the eventual equipment.
- Easier to estimate how long time the test will take.
- An opportunity to practice the test roles as test moderator or observer.

A disadvantage can be that it takes some extra time to perform with an extra test. But this time is in most cases an investment to the actual test, and in some cases it makes it faster to conduct the actual tests. (Rubin and Chisnell, 2008).

2.6.5 Develop the test plan

The test plan was based on the guidelines from the book "Handbook of Usability Testing", where all steps to create a full test plan was written down.

2.6.5.1 Why write a test plan?

A test plan is a document that targets everything about the test procedure. It is the foundation of the entire test. By using a test plan, many problems can be addressed before hand which otherwise may have been overlooked. A test plan helps the team to be synchronized and open up for communication as it includes all members of the team (Rubin and Chisnell, 2008)

2.6.5.2 Parts of the test plan

The categories of the test plan in this test was:

- Purpose, goal and objectives of the test.
- Research questions.
- Participants' characteristics.
- Method.
- Task list.
- Test environment, equipment and logistics.
- Test moderate role.
- Data to be collected and evaluation measures.
- Report content and presentation.

(Rubin and Chisnell, 2008)

2.6.5.3 Purpose, goal and objectives of the test.

This test plan addresses the how, when, where, who, why, and what of the usability test. The purpose of the test is to identify any problems with the system and measure the level of usability of the system. The test will also

give respite for eventual improvements. The goal of the usability tests is to understand if the potential users of the system can understand and use the system.

2.6.5.4 Research questions.

The research question will describe the questions that need to be resolved by conducting the usability test. The purpose of the questions is to gather relevant information of the usability of the prototype.

1. How easily do users understand how to read the charts and tables?
2. How easily do users understand how to map the charts to the tables?
3. How closely does the flow of the software reflect how the user thinks of the workflow?
4. How well do they understand the content of the topics they find?
5. Which parts of each topic do users pay attention to?

(Rubin and Chisnell, 2008).

2.6.5.5 Participants' characteristics.

The description of the participants' characteristics is important for defining the target user of the system. Research has shown that four to five test participants will expose about eighty percentages of usability deficiencies of the system for that audience. It is also shown that this audience will find all the major problems (Rubin and Chisnell, 2008). Therefore, the number of participants for the usability test was five persons. The target audience for the system is wide, but the primary users are project managers. The first questionnaire was performed on the project managers where they tested the prototype of the system. It is not optimal to conduct the final usability test on this audience, since the results can be affected when they got a glance of the system and functions. Therefore, students that are aspiring project managers are the target audience for the usability test. This audience are in the age range of 25-35 and are well familiar with the terms as Kanban, usability and affordance. Most of the students have not yet worked with similar tools, but are familiar with the theoretical background.

It is always hard to predict how many participants that are needed for the test. Some factors that affect the number are:

- The degree of confidence in the results that you require
- The number of available resources to set up and conduct the test
- The availability of the type of participants you require
- The duration of the test session

Basically, there need to be a balance between practical constraints of time and resources. Research indicates that testing four to five participants of each type

or cell will expose the vast majority of usability problems (Rubin and Chisnell, 2008).

2.6.5.6 Task list.

1. The test moderator must answer a feasibility study on the Internet before the physical test.
2. Test moderator presents the test subject to test.
3. The test person will receive an instruction of the test. Filming of the test begins.
4. The test person reads the first task.
5. The test person performs the first task. Any question and answer session with the test person containing subjective questions.
6. The test person reads the next task.
7. The test subject performs next task.
8. Number 7-8 will be repeated until all given tasks are finished.
9. After the test is finished, the test person must answer a questionnaire.

2.6.5.7 Test environment, equipment and logistics.

- All tests will be performed in an isolated test environment. The room will be set up with cameras and all equipment's to document the entire session.
- Cameras and microphones that is included in the test lab.
- A computer to conduct the test on.
- Two chairs and a table to perform the test on.
- The test will be performed on a remote computer places on Axis.
- The computer on Axis will be started with the program started.
- Nomachine installed in the computer to access the computer on Axis.
- A pen and paper to register all observations.
- The test plan.
- All scripts for conducting the tests.

2.6.5.8 Set up the test environment

The test environment needs to be tested in advance to make sure that everything is working perfectly and to eliminate any surprises. The sound, video, computers and environment itself is some things that need to be checked. Even though some error is detected and may be something that cannot be fixed, if the test

moderator is aware of the problem, the problem can then be calculated in the finish result. It may then not even be a problem.

2.2.5.9 Test moderate role.

Since this is a single-person master thesis, all roles will be assigned to one person. The test moderator will sit in the room with the test participant during the entire test session. The test moderator will first introduce the session and then perform a short background interview, which will be read from a script. After that all task will be introduced subsequently. The test moderator may ask unscripted follow-up questions if needed to clarify the test person's behaviour, comments and expectations. During the entire session the test moderator will take detailed notes and records.

2.6.5.10 Data to be collected and evaluation measures.

The questions will be answered with different types of data collected. The questions:

1. How easily do users understand how to read the charts and tables?
2. How easily do users understand how to map the charts to the tables?
3. How closely does the flow of the software reflect how the user thinks of the work flow?
4. How well do they understand the content of the topics they find?
5. Which parts of each topic do users pay attention to?

The data to be collected to each question:

- Number and percentage of tasks completed correctly with and without prompts or assistance.
- Time to complete each task.
- Count of all incorrect selections (errors).

During the test, performance data will be collected such as measures of behaviour like number of error rates, time, and counts of observed behaviour elements. Since there will be a recording of the tests, performance data can be complemented afterwards. After the test preference data will be collected, which consist more of subjective data. This data will be gathered from the debriefing session by rating scales that measure the participant's feelings and/or opinion of the system.

2.6.5.11 Tasks in the test

The task that the participant was given was based on the questions in Research questions (2.6.5.4). The task was given one at time for not causing any confusion. All tasks were written down on a manual so that all test participants were given the exact same information about the task.

2.6.5.12 Post-test questionnaire

The participants will receive a questionnaire directly after the usability test in the same room as the test was.

2.6.5.13 Debrief session

A debrief session comes in handy when more subjective data needs to be collected. While the usability test exposes problems with the system, a debriefing session can explain why these problems occur in the first place and in some cases also how to fix the problems. It is a great way to complement the usability test, since the debriefing informs the participant's point of view. The participants can explain in own words why they think an error occurred instead of the test moderator just observe and guess why (Rubin and Chisnell, 2008). The debriefing sessions was held at the same place where the usability test was performed. It was based on the post-test questionnaire where the participant rated statements, which concerned the system.

"Replay the test" Technique by the Manual method is a technique used for jogging the participant's memory during the usability test. By helping the participant remember events that occurred during the test, the participant can re-experience important emotions and reactions from the test that only him or her can explain. The Manual Method is a technique used when performing Replay the test technique. Taking notes of the participant's behaviour or event that occurred during the test and then retells the event for the participant does it (Rubin and Chisnell, 2008). Replay the test technique by using the Manual method was used during the debrief session. The entire debrief session was recorded for further analyse.

2.6.5.14 Analyse the data and observations

After conducting all the tests, the data given from the test needed to be handled. There are five thorough steps for making the most of the test results (which are described in Lo-fi prototyping 2.2.1) (Rubin and Chisnell, 2008).

2.6.5.15 Compile and summarize data

A great way of seeing patterns is to compile the data from the test into a report. Compiling the data after each test, prevents missing important details and creates even better possibilities for the next test. It is important to make it as thorough as possible which otherwise can be forgotten later on when conducting several tests. It is also important to make a copy of all the material, both videos and notes (Rubin and Chisnell, 2008). Since one person observed the tests, it was especially important to record as much as possible as quickly as possible so that nothing was missed or forgotten. Therefore, everything was compiled directly after each test. After all test were done, all data was compiled into one joined report. See appendix at the end of this master thesis to see the results. After each paragraph of text, a summary was made to get a better overview. This also made it easier to analyse the results. The paragraphs were divided into several pieces for organize the data: each task with all the questions within this task, performance data, preference data and a summary of each entire usability test. Both the performance and the preference data were then divided

into matrix for all questions. See appendix to read the results.

2.6.5.16 Analyse data

Next step after summarizing the data was to analyse the summary. Analysing the data can be very overpowering since there is a lot of information to choose from. The summary can be divided into categories:

- Identify tasks that did not meet the success criterion
- Identify User Errors and Difficulties
- Conduct a Source of Error Analysis
- Prioritize Problems
- Using Inferential Statistics

(Rubin and Chisnell, 2008).

2.6.5.17 Develop recommendations

A finding is the inference that is based on the result. Each finding should then be expanded with a discussion of what should be done, how it can be done and if it is possible to do. The discussion should also include how the change can affect the rest of the system. After making a summary of all the findings, the findings should be prioritized (Rubin and Chisnell, 2008).

2.6.5.18 The final report

The final report is supposed to include everything of the test, from the methods to the actual findings. This master thesis itself is a sort of final report from the usability tests. It includes all the steps that are taken and the result of it.

3 Technical dependencies

3.1 Python programming language

The product of this master thesis was developed in Python programming language. Python is a powerful yet flexible programming language with support for multiple programming paradigms, including object-oriented and functional programming. The language includes a rich standard library (Python, 2016).

3.2 Django

The programming was done in Django, which is a high-level Python web framework. It's used by Python programmers because of it's the easiness of developing advanced database-driven websites. Django automatically generates the administration interface from the data model in Python code, which encourages rapid development with a pragmatic design (Django, 2016)

3.3 Git

To manage source code, some sort of version control program is most commonly used. Git is very popular for Linux user since it is distributed, allowing users to create their own copy to continuing developing their own version. All versions are identified by using encryption algorithms (SHA-1). The users can send and receive changes while logging events. Even though this master thesis was a single-person project, Git was used for version handling in a fast way (Atlassian, 2016).

3.4 Gerrit

Gerrit is a tool, which is commonly used in Axis for reviewing each other's modifications on source code. Gerrit encourage the reviewer to comment, change and score the source code for improvements. It integrates closely with Git (Gerrit, 2016).

3.5 Taiga

Taiga is a project management platform for agile project management. It's a powerful tool, used for both Kanban and Scrum. It's free open source code and is free to customize for each project's needs. (Taiga, 2016).

3.6 Trouble

Trouble is a tool for handling issues, used internally in Axis, which is where it's developed as well. All departments uses the tool and an issue can be assigned to both a department, team and a person. Trouble has been used for several years and is now about to be upgraded to a new version that is more unadorned version with fewer functions.

3.7 Web developing

3.7.1 HTML (HyperText Mark-up Language)

HTML is used to define the content of web pages. In general, web pages are written in HTML and then transmitted over the Internet using HTTP. It specifies the structure of the documents and how it should be displayed (W3schools, 2016).

3.7.2 Javascript

Javascript is the programming language of HTML and the web. It is used to program the behaviour of web pages (W3schools, 2016).

3.7.3 CSS

It is used to specify the layout of web pages. CSS is describes the presentation of the HTML document and how elements must be rendered (W3schools, 2016).

3.7.4 JQuery and AJAX

JQuery is a JavaScript library that is used to simplify HTML and CSS modification, event handling and animations. AJAX is used for speeding up web development by updating a webpage without having to reload the page (W3schools, 2016).

3.7.5 Bootstrap

It is a front-end framework, which contains both HTML- and CSS-based design templates. By using Bootstrap, the user can design typography, forms, buttons, navigations and optional JavaScript extensions. It commonly used in dynamic websites (Bootstrap, 2016).

3.7.6 HighCharts

HighCharts is a charting library, which provides free open source code. The code allows for modifications as chart type, combinations and much more, It's written in JavaScript's and is based on native browser technologies, which means that the user doesn't have to install anything on the server. For using HighCharts, two files needs to run: The highCharts.js core and either the jQuery, MooTools or Prototype framework. In combination with jQuery, JavaScript's and AJAX, the charts are constantly updating with values from the server and user supplied data (HighCharts, 2016).

3.7.7 Webhook

A web hook is a user-defined HTTP call-back and a HTTP POST which will POST a message to a URL when an event occurs. When a notification occurs on a web page, Web hooks sends the custom information to the given URL. After a web application is registered to their own URLs, Web hooks are a supply way to receive valuables. Taiga supports Web hooks, so when a Webhook is registered on a project to an URL, all information about events on Taiga is sent to the given address.

4 The design process

When developing a product, the iterative process of interactions design can be divided into four basic activities:

- Establishing requirements

- Designing alternatives
- Prototyping
- Evaluation

(Rogers, Sharp and Preece, 2011). All major design requirements were first established. Then the development was divided into small iterations that were founded in these four milestones. When an iteration was done, a new life cycle began with new requirements. For establishing requirements an initial research was done in terms of reading literature, study options and analyse other studies. In parallel to the studies, analyses and identification of how issue handling was done currently as Axis. When designing alternatives, lo-fi and hi-fi prototypes was made of a simple piece of paper, which later after an evaluation was developed into a live prototype. This prototype was then evaluated in form of usability tests.

4.1 Setting up the environment

The initial step of this master thesis was to set up an environment located at the headquarter of Axis which is based in Lund.

4.2 Phase one - Initial research

The initial research was estimated to process during the first four weeks including literature study. This was done in parallel with analysis and identification of current handling of issues at Axis. The focus was on identifying the users of handling issues used today, what the where missing in their currently using tools and research what possibility this thesis could provide them. This was done by several meetings with project leaders and project managers. This was significant foundation in this master thesis.

4.2.1 Analysis and identification of current handling of issues at Axis.

There is currently one tool for handling issues internally at Axis, Trouble, which all departments' uses. It's a tool created internally at Axis. There are several other tools for handling the project management platform for agile project management. Different teams and departments use different tools, besides from Trouble, depending on availability and support. In R&D, there are still no official tools for agile development but Taiga has become quite popular.

4.2.1.1 Trouble

Trouble is the "official" tool for handling internal issues at Axis. Trouble is about to update to a new version, so by the spring of 2016 the tool will have some new features, but will in general be more stripped down. However, Trouble is not a tool for developing, but only for displaying the issues. When further

processing the issues, other tools are used. An issue in Trouble is referred to as a “ticket”.

4.2.1.2 Taiga (Kanban and Scrum)

For agile development, some departments use Taiga for project management. Taiga is never used for assigned issues for persons outside of the project. Users at Axis use Taiga for both Kanban and Scrum based projects.

4.2.2 Summary of what information that needs to be visualized

Two charts were frequently used by several teams, which were Velocity chart and Burndown chart. These two charts needed to be included in the prototype for making the tool usable for as many teams as possible since they already were using it. Then the overlap of using this tool from other tools wouldn't be so hard even if the projects were already started. Another reason for including these two charts is that both of them are proven efficient for these teams since they have been using it for a long time. So the basics of these two charts would be remained, both features for making them more efficient would be included that is optional to use.

A function that the teams currently missed was a calculation of a forecast. There wasn't any official way of calculation the forecast, some of the team just made an estimation based on experience, some used the median and some calculated by the average time in the project. Therefore, this was a functions that was desirable.

Another function that was missed was a visualization of were the bottlenecks where in the projects. It was hard to know what was the holdup for solving tasks. In many cases is was fair to make assumptions based on a hunch, but there was no actual data for it. So a function that displayed how the tasks and user stories were distributed was also a desired function, both in percentage and the number of it.

4.2.3 Analysis of Alternative graphs and charts

Before deciding to develop all charts in HighCharts, a research of other charts was done. These charts were listed with their pros and cons, to then make a decision of which tool to choose for creating charts.

4.2.2.1 Google Charts

Google chart is a free open source tool for creating charts. It was launched in 2007 and has been used diligently widely (Google charts, 2016). Pros: Support exporting images to static charts, which is an important aspect for this project since many project managers would like to share their charts with other team members. It also contains small file sizes for the libraries. It requires only a JavaScript interpreter or Internet access, which means few dependencies. Cons: It got limited configuration options (Fusioncharts, 2016).

4.2.2.2 YUI Charts

This is another powerful tool for creating charts (YUI Charts, 2016). Pros: Has been used for a while and is very robust and configurable. Cons: It is locked to flash 9, which means it can be inaccessible for all user that doesn't uses the same version. Google charts doesn't support export to image (Fusioncharts, 2016).

4.2.2.3 HighCharts

Pros: Like Google charts, HighCharts support both exporting images to static charts and requires only a Javascript interpreter. HighCharts has more customizable animation then the other two and as API support just like the other two. Cons: Misses some graphs like scrolling functions or visually editable chart. However, all these functions can be made with collaboration with other tools like Javascript (Fusioncharts, 2016).

4.2.2.4 Alternative charts for displaying the data

HighCharts provide a widely library with different types of charts. All from basic line charts to advanced combined types. Finding what types of charts that is most suitable to fulfil a purpose is more of a preference. For this master thesis, six different charts were chosen with several combinations to cover such a large audience that possible. However, if more time were being provided, the possibilities would be endless.

4.3 Phase two - Lo-fi prototyping

Based on the literature study and the current way of handling issues on R&D at Axis, lo-fi prototypes were made. The lo-fi prototypes of this thesis were drawn on a piece of paper. The first prototypes were draw and presented to a few project leaders that are currently working in projects. They have knowledge and experience of both working in agile software development but also using charts and tables. These prototypes were then evaluated by pros and cons. These lo-fi prototypes are shown in figure 4.3.1 and 4.3.2. Then a decision was made for which prototypes were going to be presented as a final lo-fi prototype.

4.3.1 First selection of charts and tables

One of the purposes of the first prototypes was to plan the navigation of the tool. Another purpose was to narrow down the selection of charts for the final lo-fi prototype. No functionality was presented on the prototype, only four different charts with realistic axes and data. The charts were drawn in colours for a better overview and easier distinguish the lines.

The first drawing presented the starting page of the tool for a first time user. It showed a white and empty piece of paper with a squared shaped button with a plus sign on. The second and third piece of paper showed four different charts on. Every chart got its own square button with a cross on it for deleting a chart. Even though the prototype didn't provide any physical functions, it provided a

glance of what the system could look like and it also made it easier to determine usefulness and feasibility without having spent too much time.

Figure 4.3.1 shows a lo-fi prototype of a combination of four different types of charts. The first one (on top to left) shows User stories statistics that is measured number of user stories per hour. In this example there is four different statuses. There is a drop-down button for selecting project, a drop-down button for changing type of chart and a button marked with "X" to delete current chart. The next chart (top to the right) is a Pie chart that has the same buttons but displays the total percentage of time spent by user stories in different states. In this example there is four different statuses. The third example (bottom to left) displays a chart that shows ticket statistics. The different between the first one and this one is that the first one displays statistics for user stories and the other one displays tickets. The last one (bottom to the right) displays a speedometer that shows the current speed of number of ticket or user stories that is solved per day. This chart displays the same buttons as the other ones.

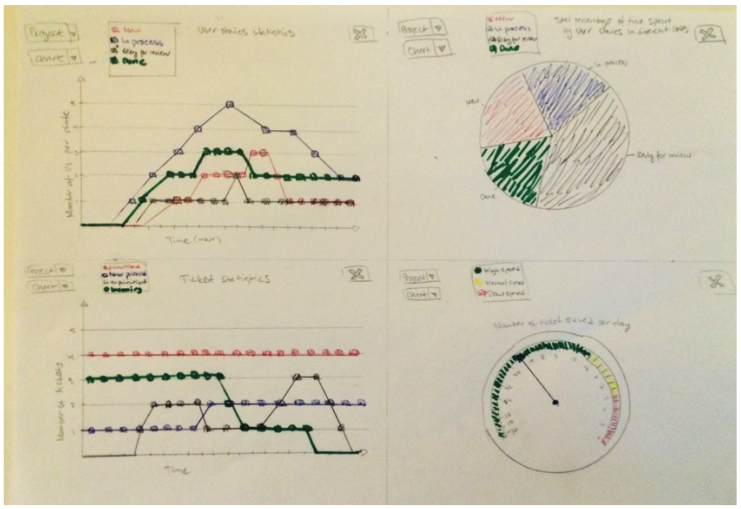

Figure 4.3.1 displays the first lo-fi prototype.

The first chart in the next lo-fi prototype, shown in figure 4.3.2 has the similar statistics chart as the previous chart but has no buttons for changing project or type of chart. Instead there is a drop-down menu for changing what interval that is preferred. When changing the interval, the value of the x-axis is changed to the chosen value. The second chart on the next figure is a velocity chart, which is regularly used by several teams on Axis. There are no buttons or functions. The y-axis displays the number of story point. The x-axis displays all the recent columns in pairs, one for the committed story points and one for the actual finished story points. The third and fourth chart displays a pie chart and a Burndown chart. The pie chart is the same as the one in the first prototype, but without any functions or buttons. Several teams at Axis regularly use the Burndown chart. The x-axis displays the time and the y-axis displays the remaining time elapsed. The linear line across the chart is the actual Burndown. To the left in the figure the user can read out the remaining values, which in

this case is 50 and the end date, which in this case is 1 of January. The user can then choose to change the end date, which will simulate the Burndown line to change its end date. The last chart is a Pareto chart. The x-axis displays all the status that is set in this project and the y-axis to the left displays the time. To the right the user can read where the eighty percentages is which statuses that are included in this eighty percentage.

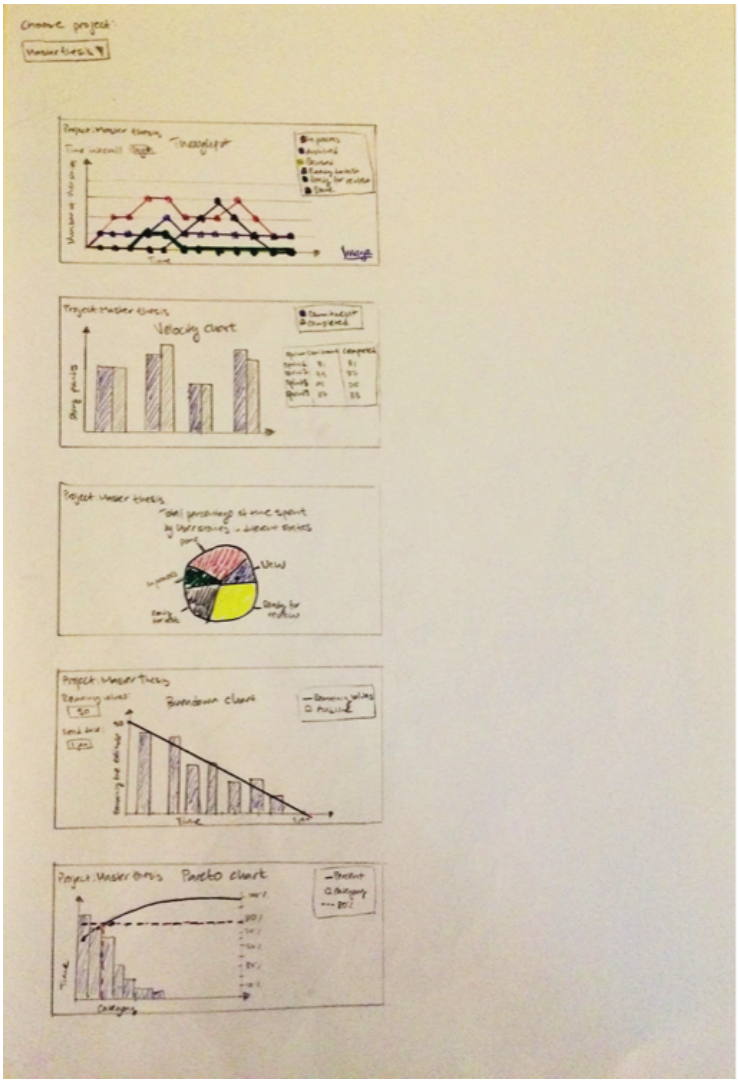

Figure 4.3.2 displays the second lo-fi prototype.

4.3.2 Second selection of charts and tables

Together with two project managers that currently work at Axis, an evaluation was made to decide which charts were useful in projects. There wasn't time enough to implement all of the charts, and therefore some of them were

deselected. Even though the main activity in this master thesis is to measure usability in streamlining the handling in R&D project, there is still an interest in making this tool usable at Axis in projects. There is a possibility to implement all of the charts, but because of the time limit, there had to be some sifting for this master thesis. Therefore, some of the charts were selected to be implemented and some of them were deselected for future activities. After a selection was made, six charts were chosen. The six charts were then expanded with functions, buttons and somewhat more details. But they were still lo-fi prototypes that later were going to be used as evaluation material for the final prototype.

The first final lo-fi chart is shown in the next figure, which is named 4.3.3. This is a cumulative chart and this chart wasn't currently used at Axis in any of the polled teams. The x-axis shows the dates and the y-axis displays the number of user stories. This chart would be useful for making forecast, which is based on data from the project. Two buttons for start and finish date of the forecast, which is the on to the left at the top of the figure. This buttons are used for changing which date to be displayed in the chart. The lower button to the left side is a sliding button, which changes how many days to be displayed on the forecast. In the middle of the chart, there is a dotted line, which displays the current date. The date before the dotted line is the history of the data at this project since the start date. All data after the set end date is displayed in the chart to the right of the dotted line. Different colours sort all status.

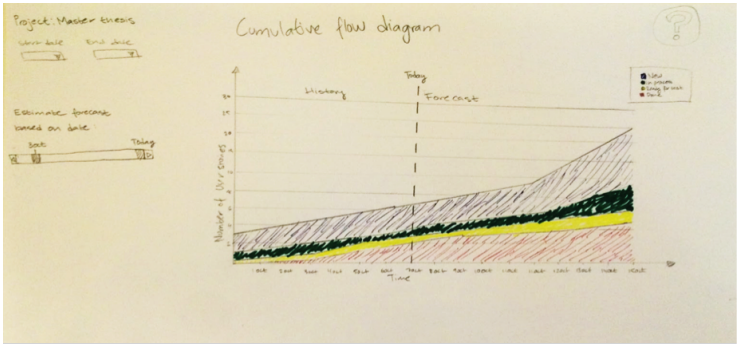

Figure 4.3.3 displays a lo-fi prototype of a Cumulative chart.

The figure 4.3.4 is a lo-fi prototype of a Velocity chart. Compared to Velocity chart in the figure 4.3.3 this is combined with buttons and tables. There are two buttons to the left in the chart, one for setting the start sprint to be displayed and one for the end sprint. There is also a table under the chart listing all the sprints in order, with the average sprint listed first. The average sprint shows both the committed points and the average completed sprint. The average sprint is based on the chosen start and end sprint and would be helpful for the user to see how accurate the committed sprints have been and how the next sprint probably would be.

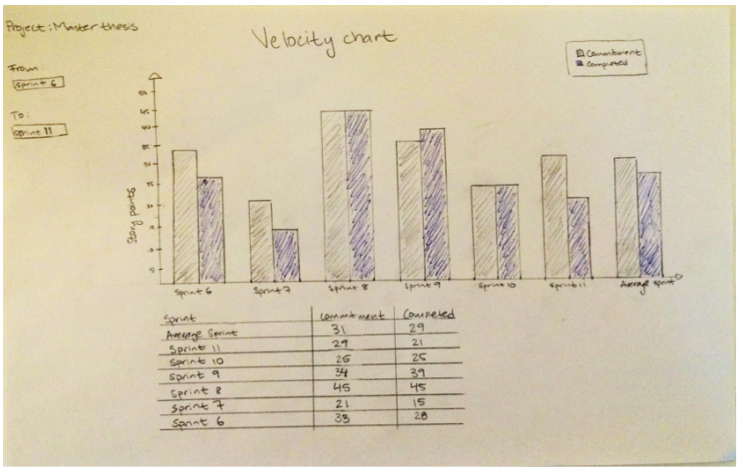

Figure 4.3.4 displays a lo-fi prototype of a Velocity chart.

The next lo-fi prototype is figure 4.3.5, which is a mixed chart with both a pie chart and a throughput chart. The pie chart is the same pie chart as in figure 4.3.3 with all user stories divided into status, which is marked with different colours. The throughput, which is the chart to the left in the figure, displays the elapsed time by the number of user stories per state. The table underneath the two charts shows all states in the project divided into status. The status shows the statistic development of how the percentage of user stories changes state. The statuses are then divided into different time intervals and at the end a forecast, which is based on the development in history of the project. This chart would be useful to display how the tasks and user stories are distributed over the status and make it easier to predict the bottlenecks.

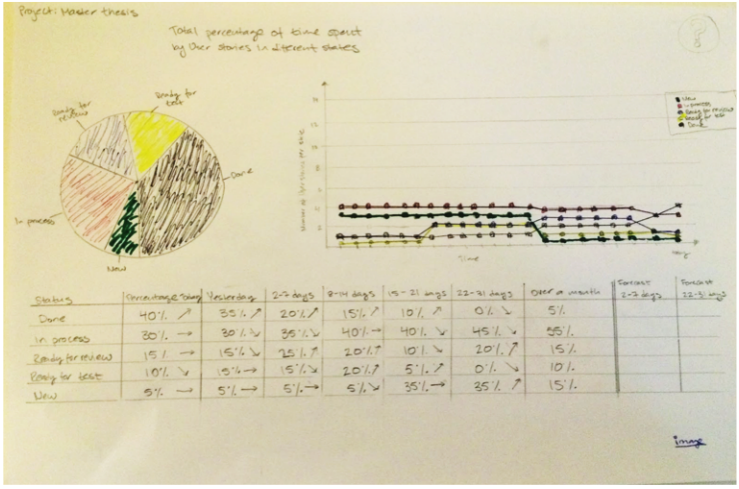

Figure 4.3.5 displays a lo-fi prototype of a combination of a Pie chart and a Throughput chart.

The prototype shown in figure 4.3.6 shows two charts and a table. The charts are a pie chart and a throughput for only one state. There are also two buttons

for changing state and the number of days to be displayed. The table underneath the charts shows information for each the user story in the chosen state. Like the previous chart, this chart would be useful to display how the tasks and user stories are distributed over the status and make it easier to predict the bottlenecks over one status at time.

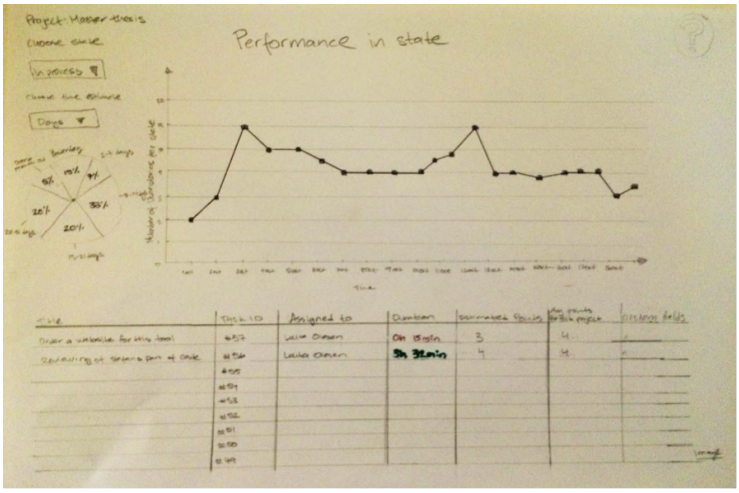

Figure 4.3.6 displays a lo-fi prototype of a Performance per state chart.

The next prototype is shown in figure 4.3.7. The prototype contains a classical Burndown chart with bars and a line to display the actual Burndown. The actual Burndown is based on which end date the user selects to the left of the figure. The rate of productivity is based on the number of days from start of the Burndown to the chosen end date divided on the man-hours per day. However, what's new in this chart is a comparative Burndown chart, which is a second line that is independent from the other line. The second line is based on the parameters from underneath the first parameters of the actual Burndown. There, the user can chose the start and end date of the Burndown or manually set the rate of productivity. Both the actual and the comparative Burndown lines will then be displayed for the user to compare the actual Burndown to a fictive one. This would make it easier for project managers and projects leaders to calculate how many man-hours is needed to meet a deadline. The table under the chart displays information if the user stories and tasks.

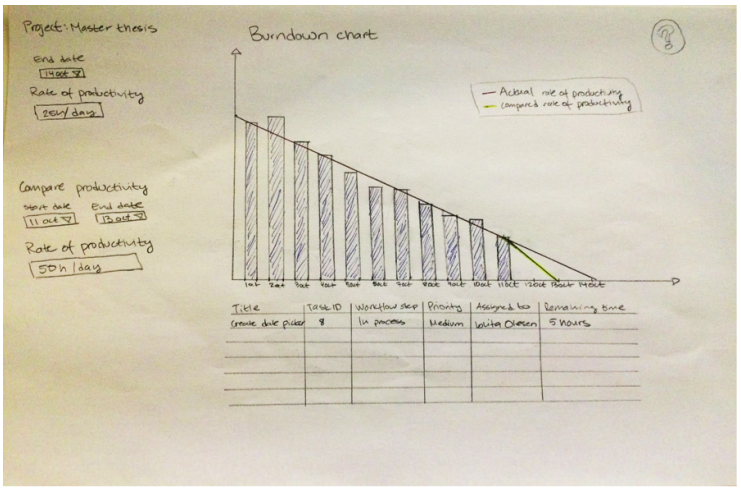

Figure 4.3.7 displays a lo-fi prototype of a Burndown chart.

The last lo-fi prototype is shown in the next figure, 4.3.8. It is the same as the Pareto chart in the in figure 4.3.2. The only different from the other lo-fi prototype is that this one has a table underneath which displays the status, the frequency and the cumulative frequency, both in percentage, for each status.

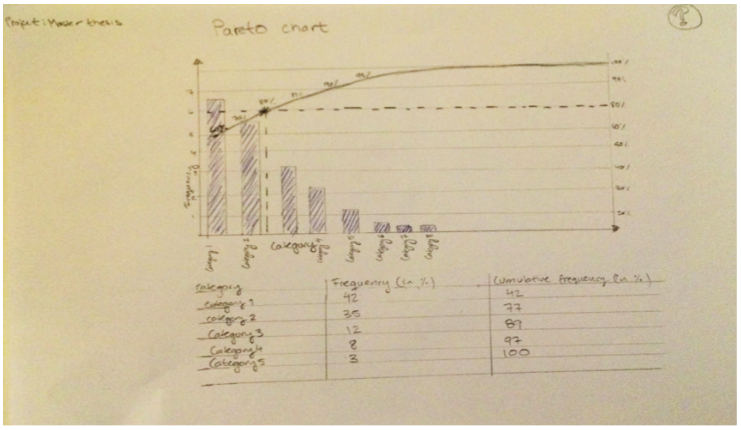

Figure 4.3.8 displays a lo-fi prototype of a Pareto chart.

4.4 Phase three - Usability testing

A usability testing was conducted on the Lo-fi prototypes. The test participants were project managers and project leaders that uses similar products in their daily work and the number of participants was ten people. The agenda of the meeting was as follows:

- Background history of my master thesis and me. I presented my master thesis and who I was.
- A list of what was missing in today's tools. I summarized the tools that

they used today and what they were missing.

- A summary of my findings within the company.
- Each prototype with a short description of the charts.
- Questions from the participants and suggestion to be improved.
- SUS questionnaire that was filled in anonymously.

4.4.1 System Usability Scale

For grading the usability a SUS (System Usability Scale) was used. The SUS questionnaire was based on ten different questions. The questions were based on a template given in Usability Handbook. (Rubin, J. and Chisnell, D., 2008).

1. I could imagine using this tool frequently in my daily work.
2. I think the tool seems unnecessarily complex.
3. I think the tool seems easy to use.
4. I think I would need assistant to use the tool.
5. I think that the functions of the tool are relevant to the tool.
6. I think it is too much inconsistency in the tool.
7. I think that users can learn relatively quickly to use the tool.
8. I think the tool seems awkward to use.
9. I think the tool can help me in my daily work.
10. I have to learn a lot before I can start using the tool.
11. I think the features are missing, namely:

All ten participants filled in the questionnaire anonymously. All questions were ranged from 1-5 where 1 is "Does not agree at all "and 5 is "Absolutely agree". The results from the questionnaire can be found in figure "SUS questionnaire of lo-fi prototype" in 4.4.1.

4.5 Test result and summary from lo-fi test

Figure 4.5 displays the results of the lo-fi prototype. All questions are range from 1-5, where 1 is "Does not agree at all" and 5 is "Absolutely agree". All questions are attached in appendix.

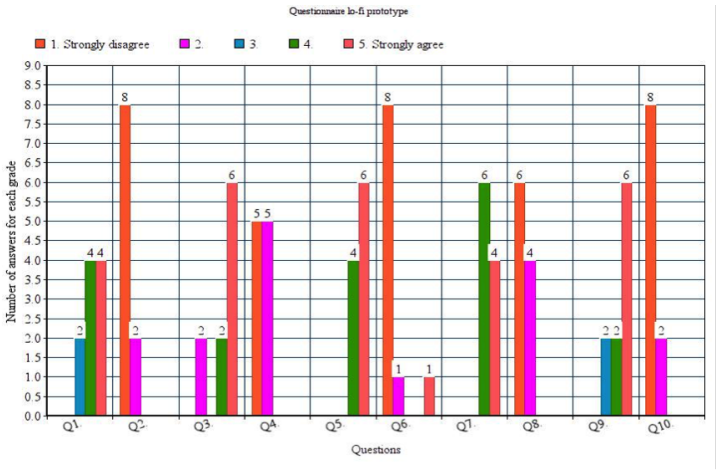

Figure 4.5 displays the results from the SUS questionnaire of the lo-fi prototype.

The answers to the questionnaire is then summarized and graded to measure the usability of the prototype. If the average grade from the questionnaire is above 68 points, then the system has a high usability. The point that the questionnaire from this lo-fi prototype was 87.5, which is then considered a high score. To see how the calculations are made, please refer to 3.3 System Usability Scale.

The positive aspects were that most of the participants considered the prototype relevant for their daily work and thought that is would increase the efficiency of their projects. The negative aspects were that they needed quite specific functions and chart for replacing it from the tools they used currently. However, just because they couldn't replace could depend on that they already had invest time in the current tool for a specific project. But that did not mean that they didn't considered to use this prototype in future projects from the start. Then they may not need other tools as a complementary.

All data was collected and a summary of the questionnaire was made. A list of what was prioritized was then created based on what was relevant for this master thesis and what was relativistic keeping up with temporal. It was also important to know what the test participants thought was important for their daily work and what was missing, that still was important for using it.

4.6 Phase four - Hi-fi prototyping

The hi-fi prototype needed to be as real as a finished product as possible. The goal of the prototype was to actually implement it and make it usable for basic projects. A fictive project was created and registered in the hi-fi prototype for two reasons. First, it was important to test the tool on a project that was a realistic project to find bugs and to get a feeling of how it's supposed to be used. Even though all parts isn't implemented, buttons and links still looked live and has some sorts of function. Figures 4.6.1- 4.6.6 displays screenshots from the hi-fi prototype. Figure 4.6.1 displays the first page of the hi-fi prototype. There is only one project that is currently registered at this figure, named

239. The user can choose to display project 239 in seven different types of charts.

Figure 4.6.1 displays the first page of the hi-fi prototype.

Figure 4.6.2 displays the “Status count chart” for project 239. The x-axis is a time line, which is set to days and the different colour represent different types of status. The y-axis is set to the number of User stories per state. The blue button is used to download the figure as it is currently. This blue button has the same function at every chart.

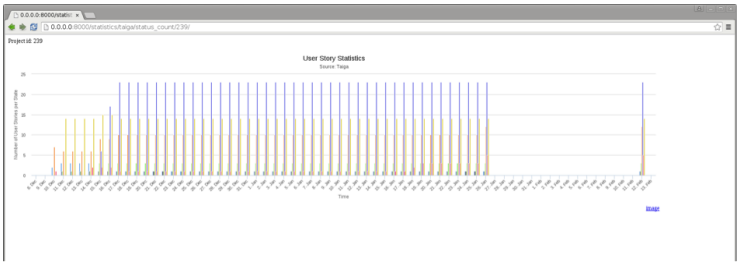

Figure 4.6.2 displays a ”Status count chart”.

Figure 4.6.3 displays a “Pie chart” for project 239. The different colours represent the percentage of each issues distributed of all status. When pressing the checkbox “Include status that are “closed”, the users can chose if they would like to include the state “closed” of not.

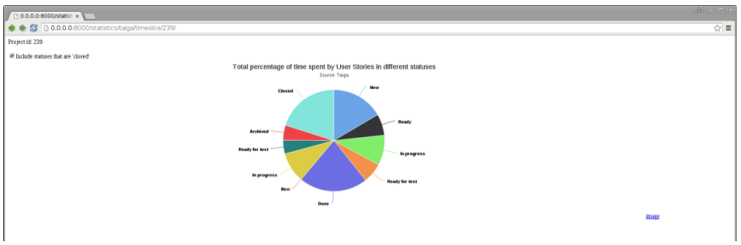

Figure 4.6.3 displays a ”Pie chart”.

Figure 4.6.4 displays a “Burndown chart” for project 239. The x-axis is a time line, which is set to days. The y-axis is set to the number of User stories per state. The different colours represent the different statuses. All states per day represent one bar, and the colours divide the statuses. The user can chose if they would like to include different status or not by pressing the coloured checkbox. By default all checkboxes are chosen. By pointing at one of the bars in the chart, a pop-up box is displayed with a summary for all status at chosen date.

The first text field in to the left in the figure is used for setting the start date for the Burndown. When pressing a date in the text field, a green line will appear starting at the chosen starting date in the x-axis and on the top of the bar of that day. The end date will then by default be the date of today. The next text field is used to change the end date. The line will then end at the x-axis at the chosen end date. Under the end date text filed is the rate of productivity typed, which calculate how many user stories per day the user needs to solve for managing the Burndowns date that are chosen. This is based on the number of User stories that are at the starting date. The following two text field is named "Comparative Burndown". They have the same functions as the two text fields above, and will show a new line in red. This is used for comparing the line to the other line. Under the end text field is typed the rate of productivity for the comparative Burndown line. The table at figure 4.6.4 lists all the User stories and tasks in this project. The User stories are marked by making the text bold. All tasks are placed underneath the User story that they belongs to.

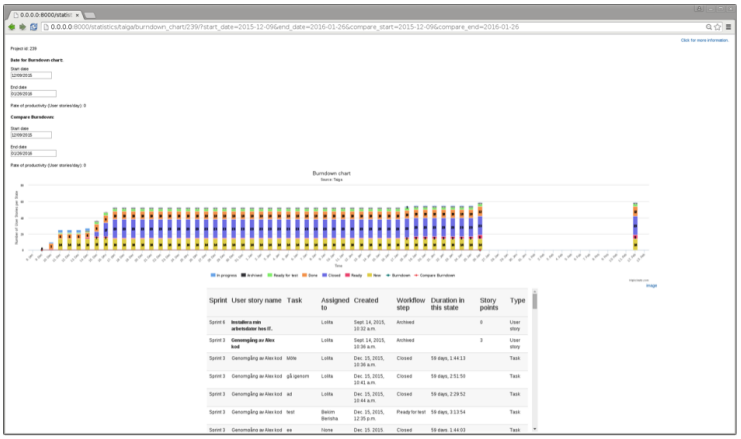

Figure 4.6.4 displays a "Burndown chart".

Figure 4.6.5 displays an "All status chart" for project 239. The x-axis is a time line, which is set to days and the different colour represent different types of status. The y-axis is set to the number of User stories per state. Just like in other charts, the user can chose if they would like to include different status or not by pressing the coloured checkbox. By default all checkboxes are chosen. By pointing at one of the bars in the chart, a pop-up box is displayed with a summary for all status at chosen date. A difference from "Burndown chart" is that all statuses are represented in its own bar. So in one day, several bars are displayed next to each other. Like figure 4.6.3, this chart has the functions for a Burndown line and a comparative line as well. The table at figure 4.6.5 lists all the User stories and tasks in this project. The User stories are marked bold. All tasks are placed underneath the User story that they belong to.

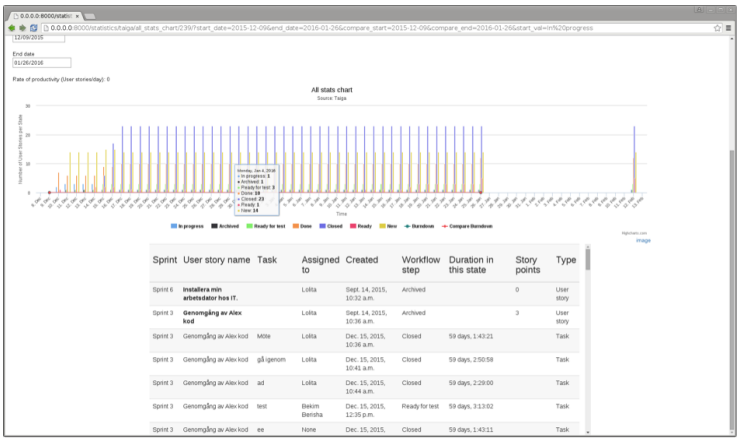

Figure 4.6.5 displays a "All status chart".

The last chart, figure 4.6.6, displays a "Cumulative chart" for project 239. The x-axis is a time line, which is set to days and the different colour represent different types of status. The y-axis is set to the number of User stories per state. . Just like in other charts, the user can chose if they would like to include different status or not by pressing the coloured checkbox. By default all checkboxes are chosen. Anywhere the chart displays a pop-up box, which displays a summary for all status at chosen date. The vertical green line in the chart is marked at the date of today. Everything after that green line is a forecast. The forecast is based on the parameters from the text fields. The first text filed is a start date and the next is the end date. The next function is a button with a dropdown function, which is named "Type of estimation". The user can chose between the values median and average. Next button is named "Period of time" and the alternatives is displayed in a dropdown window with the values 1 day, 1 week, 1 moth and a year. The forecast is then calculated with all these parameters. All history of the User stories within the end and start date is then calculated to either an average value or the median of it and is plotted as a forecast for as long as the chosen period of time.

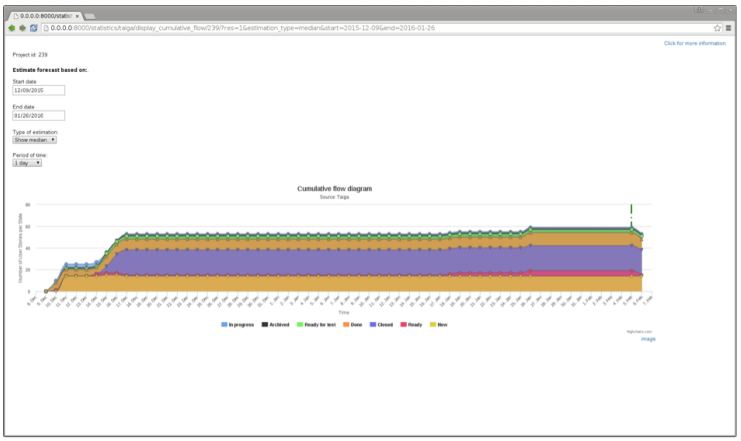

Figure 4.6.6 displays a "Cumulative chart".

4.7 Writing a test plan

The test plan addressed the how, when, where, who, why, and what of the usability test. The purpose of the test was to identify any problems with the system and measure the level of usability of the system. The test was also intended to give respite for eventual improvements. The complete test plan is attached at the end of this master thesis.

4.8 Phase five - Conducting the Usability test

4.8.1 Finding the test participants

The goal was to conduct the final usability test on five test participants. The test was estimate to take about 30 minutes to conduct. The number of five participants is reasonable to gather enough information of since it would be enough to make assumptions of the results (see chapter 2.6.5.5). The initial test on to lo-fi prototype was made on 10 participants, both project managers and project leaders. If they had conducted the second test as well, they would already have a prior knowledge of the tool, which could have affected the final test results. Therefore the test needed to be tested on fresh participants that haven't seen the tool before. Since students of Civil engineering on Computer science has some basic knowledge of similar tools and has most likely heard some keywords as Kanban, Scrum and different kinds of charts, they were an excellent target group. Therefore, all five test participants was students of LTH, with the age between 25-30 years old. All of them are currently studying to become civil engineers and at their final year. There was also a pilot test conducted on a test person that wasn't included in the rest result. This person was instructed to perform all given task that was included in the test to find out if anything in the instructions was unclear or vaguely described. This made also made it easier to estimate how long time the test would take.

4.8.2 Prepare the test environment

The test was conducted in a secure test environment at IKDC, Lund, with a full equipment of sound and video systems. This environment was isolated so there weren't any distractions. The room was soundproofed and the only door into the room was marked with a "Do not disturb" sign. All test equipment was checked before the first test participant arrived. Since the tool was an internal tool for Axis, a SSH connection to a stationary computer at Axis was needed to access the tool. Then a laptop was used as a testing tool to perform the test on.

4.8.3 Performing the tests

All tests were performed one by one. The average test took about 20 minutes to conduct. The test participants were lead into the test room, and then given the introduction to the test. The introduction is attached at the end of this master

thesis. Then all cameras were started to record the test. The test participants were told to read out loud each task and try to think out loud while they performed the tasks. One camera recorded the participants to see reactions and another one recorded the screen at the computer. After all tasks were done, a short debriefing was made to gather any questions and information of the participant's feelings and reactions of the tool and the test. This was done in the same room directly after the test. The laptop was still running while the debriefing took place, for helping the participants to jog their memory according to "Replay the test" technique. After the debriefing, the test participants were given a post-test questionnaire to fill in.

4.8.4 System Usability Scale

As written above, all test participants were given a questionnaire to measure the usability of the system. This was the same questionnaire that the participants of the test of the lo-fi prototype were given. (All questions are written down under 2.6.3 Usability testing - System Usability Scale).

5 Results

5.1 Test result for final test

Figure 5.1.1 displays the time elapsed for each participant for each task. There were five participants, which represent a given colour.

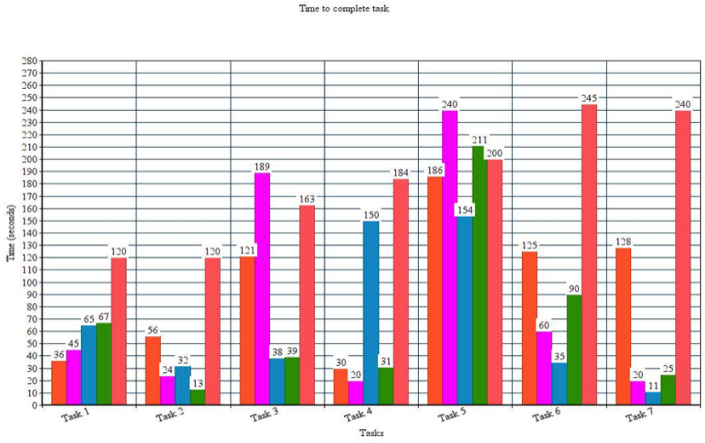

Figure 5.1.1. Time elapsed for each task for each participant. The task is measured in seconds.

See appendix for finding out the tasks. Each task represents a scenario. Next figure, 5.1.2 displays the number of errors with and without prompts or assistance.

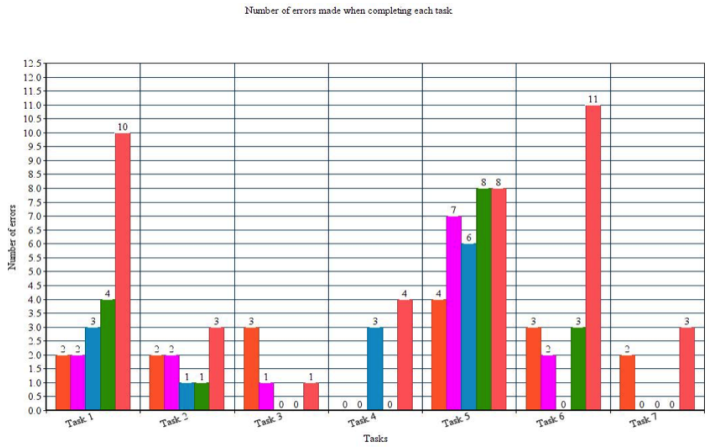

Figure 5.1.2.Chart displays the number of errors made per test participant per task given.

Figure 5.1.3 displays the results of the hi-fi prototype. All questions are range from 1-5, where 1 is "Does not agree at all" and 5 is "Absolutely agree". All questions are attached in appendix.

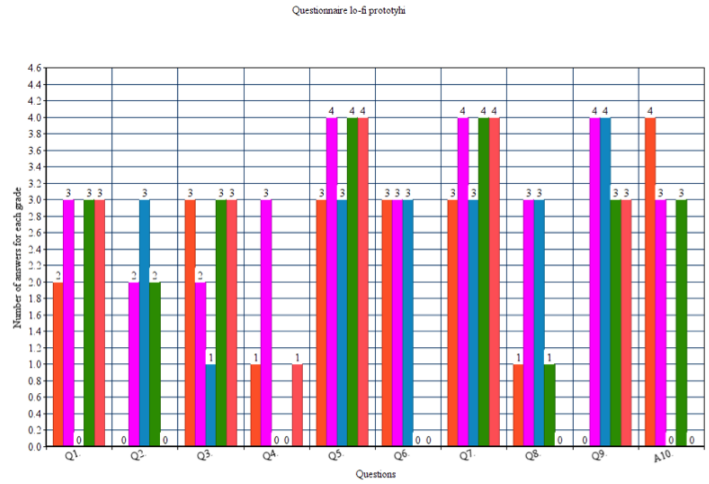

Figure 5.1.3 displays the result of the SUS questionnaire of the hi-fi prototype.

The answers to the questionnaire is then summarized and graded to measure the usability of the prototype. The average score from the questionnaire was 60.5 points. Most of the test participant wrote on the questionnaire that the reason why they gave low score was mainly because it was too slow for using daily. And also, since they were not currently working, then wouldn't know if they would need the tool in the future.

5.2 Task 1

Task 1 was to find out the number of User stories under the status "Done" for a specific date. This was supposed to be done in a Throughput chart, that didn't have much functions, text or feedback. The average time to complete the assignment was 66.6 seconds. The difficulty of this assign wasn't the assign itself but was somewhat confusion since there was a delay of the Internet access. The average number of errors in this task was 10 errors. All of the participants made the same error, which was that they didn't look in the correct bar, but all of them looked at the correct place directly.

5.3 Task 2

The next task was to find out the number of User stories under the status "New" for a specific date. Compared to the first task, this was supposed to be done in a different chart from the first one. The difference was that there were more functions like an explanation of what each colour represented and a summary for all states no matter which status or bar the user choose. The average time to complete this task was 49 seconds. The average number of errors made in this task was two.

5.4 Task 3

In task number 3 the participants were asked to make a forecast in the cumulative chart based on a given start and end date. The participants were asked to make a forecast based on a median value. The average time of completing this task was 110 seconds. The average number of errors made in this task was one, where errors was misunderstanding the task and setting wrong dates.

5.5 Task 4

The fourth task was to create a Burndown chart based on a start and an end date. The average time to complete this task was 83 seconds. Two participants struggled with this task since the Internet access worked poorly. In this task the average number of errors was one, where all the errors was made by two of the participants.

5.6 Task 5

Task number five was an extension of the last one. The users were going to save their last result and then create a comparable Burndown in the same chart as the last task. The new Burndown should start the same date as the actual Burndown but finish a date when all bars was underneath the new line. The average time for completing this task was 198 seconds, which was the highest result. There was a lot of confusion of how the task should be solved and several

mistakes were made. The average number of errors was seven in this task, which was the highest number of errors in the test.

5.7 Task 6

The next task was to find out how long a user story has been in the same status. The user was going to search in a table on the page "All stats". The table was placed underneath the chart. The average time for completing this task was 111 seconds. The average number of errors in this task was nine and was divided differently among the participants. All of the participants made errors that were directly dependent on the delay of the internet. One of them didn't find the table at first and didn't know where to look for the information.

5.8 Task 7

Task number seven was the last task in the usability test. The task was to search in the same table as task number six but for different information. This time the participant was supposed to count the number of tasks belonging to a specific user story. The average time for completing this task was 84 seconds. In this task, the average number of errors made was one, where only two of the participants made them. The other three completed the task correctly without making any errors.

5.9 Summary of what is missing

The final product of this master thesis was a hi-fi prototype. This means that it is a product that is implemented but is not ready for use. If more time was given, more charts would have been implemented for better usage of the tool. It would have been nice to extend features of making forecasts, as for example more advanced mathematical analyses that can track trends and estimate a better forecast.

Another thing that was missing was a better support for several browsers. This was something that came to notice during the usability tests. For example, the date button didn't display a full calendar. Instead the user had to type the date manually. Another thing was that the table's header didn't show when the user started to scroll.

If more time was given, the implementation would have been test-driven. This would have been making the tool much more reliable and easier for others to extend the tool. This, however, would have taken much more time to perform.

At this time, the tool is only usable for users that are currently using Trouble and Taiga at Axis. At the company, several teams use other tools for handling issues. In the future, there is a possibility to extend the access to more tools.

Taiga uses webhooks for pulling out information of events. This is not the optimal tool for handling this kind of information. Since the client has to ask the server if there are any changes, there is always a possibility that the client

asks the server too late and information gets lost. A better way to implement this would have been if the client and server uses both push and pull actions so that no information would have been lost. The safest way would have been if the client asks the server for information every time it's needed and the server pushed information every time there is an event to be registered. It wouldn't be necessary to use webhook for the implementation of the hi-fi prototype. But it would be more time-consuming to implement it differently.

6 Discussion

6.1 Process

The timeline that was set at the start of this master thesis was extended with two weeks, which still is reasonable time estimation. All steps in the process have been conducted within the time estimate, and have been documented in a logbook. This kind of prototype is never fully completed, there is always something more to add, remove or change. The implementation of the prototype was the most time-consuming part. The ideal was to spend most of the time on the tests and the theoretical design. However, the result wouldn't be as accurate if the functions would have been just theoretical.

6.2 Result

6.2.1 The SUS questionnaire for the lo-fi prototype

The lo-fi usability test got 87.5 points in score from the questionnaire. A score above 68 point is considered a good usability. Therefore a decision was made to continue the work based on some of the lo-fi prototypes. If the score had been lower, there would be a need for some more drastic changes. The questions that were given a lower score were taken into consideration, like the preferred if it was easier to use and there is some inconsistency in the tool. The two participants that did not think they could use the tool daily also made a comment that the reason that they didn't think so because they didn't currently use Taiga, which the tool is currently dependent on.

All final usability tests were conducted in a usability laboratory in IKDC, Lund. The benefits of conducting the tests there was it is a controlled test environment. All outer parameter that can affect the test, are under control and was therefore easier to analyze. However, when conducting the tests, all test persons had some trouble with the Internet access and the tool itself was very slow. This was based on two reasons:

1. When connecting via SSH to your server, the traffic have to travel a longer distance and all events take longer to respond.
2. The Internet access in the Usability laboratory has a low bandwidth and is therefore responding slowly.

Unfortunately the combination of these two parameters resulted in a very slow response in total. This problem resulted in some confusion and some frustration from the test participants. Since some interaction was delayed with a few seconds compared to the other tests sessions, it affected both the physical time to execute a task but also caused a misleading of the participants first reactions. An example of this is when reading the number of User stories in this interface, the participant had to place the computer's mouse on top of the correct bar in the chart for displaying the pop-up label. However, since there wasn't an immediate interaction, the participants felt that they were doing wrong and moved the mouse before they could see the number. So even if the first intuition was immediately correct, the executions resulted wrong since there was no immediate feedback.
Before conducting the real test, a test pilot tried the test on the same laptop to try out the tasks and to see the response through the SSH connection. Outside of the lab, the connection worked just fine with some delay. The Internet access was also evaluated beforehand on another computer, and worked just fine with some delay as well. But the combination of the SSH connection and the already slow Internet access in the laboratory made it worse. The first four tests worked just fine, with some delay in some of the tasks, and therefore the test continued on the same computer. During last test the Internet access got worse and therefore resulted in remarkable worse result. At one point, the connection had to be closed and then started again and got better. All tasks could have been executed faster in a better Internet access was given. However, the first reactions was recorded, and all buffering time was excluded from the results in figure "Time result of hi-fi prototype" and number of error made per task is shown in figure "Count errors from result of hi-fi prototype" in 5.8.2 Test result for final test.

6.2.2 The Task from the final usability test

6.2.2.1 Task 1

The average time to complete this assign was 66.6 seconds. This is not a remarkably long time for completing this type of assign. When conducting the tests, the last test person had some trouble with the Internet access and the tool itself was perceived as very slow. Since there wasn't an immediate feedback, the participant felt that she was doing wrong and moved the mouse before she could see the number. After several attempts, she finally saw the pop-up label, and read out the number without having to ask for help. So the intuition was immediately correct, but since there was no immediate feedback it resulted in confusion. But the first reactions of was to place the mouse on top of the stable, which was the correct way to find out the information to complete the task.
The average number of errors in this task was 10 errors. The number seems quite high based on the simple task. But because of the delay of the feedback, the textbox with the information didn't appear right away and therefore caused some confusion from the participants if they were looking in the right place. They were all performing the task correctly right away, but were misled because of the delay and started to question themselves.

6.2.2.2 Task 2

The average time to complete this task was 49 seconds, which is a lower average of time than the last one that was 66 seconds. This is however somewhat misleading because the first participants didn't read out the instructions correctly. Instead on reading out the number of user stories from one status at one date, the participant summarized all status at that same date. When the participant thought he was done, he was informed that the assignment was to find out for only one status. Then he knew the answer directly. It took 32 seconds before he realized that he didn't solve the task correctly. If the time that he was solving the wrong task were excluded from his result, the average time would be 42 seconds. The last participants had some trouble with the internet access. When she held the mouse on top of the correct bar, the feedback was delayed and the information didn't appear right away. Before the information appeared, she moved the mouse and the information didn't appear at all. By the time she realized that she was doing correctly all along for finding out the information, almost 40 seconds has passed. If these 40 seconds were excluded from her result together with the exuding from the first participant, the average time would be 34 seconds. 34 seconds compared to 66 seconds is a huge difference. It is clear to say that the participants showed that it was easier to complete the task if all status was shown no matter what bar that was chosen. It may also be easier to complete the task because of the colour of all states was given, which all test participants confirmed in the debriefings afterwards.
The average number of errors made in this task was two. One of the participants thought that he needed to count all the status combined, but this error was made because of he wasn't reading the instructions properly. All the others made the mistake that they looked at the same date as was given in the last task, all because they didn't read the instructions properly.

6.2.2.3 Task 3

The average time to complete this task was 110 seconds. All the participants directly knew where to set the start and end date. Three of the participant's didn't realize directly that they could make the forecast for a month with the help of a button. They thought they needed to zoom in the dates and tried to calculate the median manually. This was of course very time consuming, and therefore the average time was so high. When the three participants thought they were done with the task the test moderator said: "The answer is not correct, and you don't need to calculate manually. Have you tried all buttons to complete the task?" Then all of them solved the task in a few seconds. So looking back at the recordings, if you exclude the part when they tried to calculate the answer manually, then the average time would be 27 seconds. However, it was interesting that more than half the test group missed the button. It could be either that the task description was vague, the tool was poorly designed or a combination of both. In the debriefing, the three test participant that missed the button was asked about them. They all said that they didn't understand how to make a forecast and didn't look very closely after a function to help them. If the line of the date would have been more visible, so that the user sees that the forecast will appear after the date of today, then it would be clearer that the forecast will be in the same chart and that the user wouldn't have to calculate it manually.

The average number of errors made in this task was one. Two of the participants did not make any errors at all, two of them made two errors each and the last one made three. One of the participant thought that he needed to calculate his own forecast. The other error that was made was that they missed that one set the wrong date and the other one didn't realize that she made correct and changed again to a wrong thing because of the delay.

6.2.2.4 Task 4

The fourth task was to create a Burndown chart. The average time for completing this task was 83 seconds. In this task, two of the participants have some trouble with the access to Internet, and it took a longer time for them to complete the task. Still, the entire time to complete the task was included in the result because they wasn't really sure if they were doing the right thing when they pushed different buttons. All of them did the correct thing by choosing the date from the date function. However the two of them that had the longer time result were still unsure if they completed the task and continued to look for functions. In the debrief session, the participants were asked about there actions. The two answered that they wasn't sure if they had completed the task correctly since there was no feedback of doing so. This confusion probably wouldn't appear if the access of Internet had been working better. If this problem was excluded from the result, the average time would have been 28 seconds. In this task the average number of errors was one, where two of the participants and none by the three others made all the errors. The error was because of the delay, yet again. The participants thought they was solving the task wrongfully because of the lack of response in the tool and was changing the answer several times before realising that they were doing the right thing all along.

6.2.2.5 Task 5

Task number five asked the participants to make a comparable Burndown chart with two lines, one with the actual Burndown from the last task and one for estimating. This was clearly the most difficult task to complete, with the average time of 198 seconds. There was a lot of confusion of how the task was going to be solved and by the look of the participant's faces and body language, you could see that they were a bit stressed with completing the task compared to the other tasks. All of the test participants asked the test moderator several questions about how they were going to solve the task. The questions were:

- How am I supposed to know how to complete the task?
- Do I need to keep the other Burndown?
- Am I doing the right thing now?
- I don't know what a "Burndown" is, what is it?

A Burndown was shortly explained at the end of the task description, but was overlooked by all of the participants. When they started to ask questions, they were reminded to read the complete task description, and could then complete the task without any further questions.
Compared to the other tasks, there was a lot of information on this chart. However, it was the same information at the page for the previous task, but

none of the participants had any trouble with completing this task. And even when they were solving the task correctly, they were only increasing one day at time. Totally, they needed to increase with three weeks from the last Burndown, and therefore the task took much longer than the others ones. If the time that they were doing the right behaviour but not finding the correct date were excluded from the total time, the average would be 83 seconds. That is a huge time difference, but that also includes some confusion. In the debriefing the participants was reminded of their questions that they had during the test. They all wanted a better explanation of what a Burndown was and didn't realize that they were going to use two different types of lines. This however was supposed to be the hardest task since there was a lot of information and demanded some knowledge of how a Burndown worked. The information explaining what a Burndown is should have been placed at the beginning of the task description, so that the participants wouldn't have missed it. That would have decreased the confusion and would have made the task easier to solve from the start.
Some of the participants said during the debriefing that they felt that there was a lot of information in the tool at the same time and it took some time before they realized where to look for the correct information.
The average number of errors was seven in this task, which was the highest number of errors in the test. The time for completing the task was the highest as well, which indicates that the number of error and time to complete was depending on each other. As written above, there was a lot of confusion for completing this task. The main reason for the confusion of what a Burndown was. This information was given at the end of the task, which should have been given at the beginning of the task. So, the conclusion is that the task itself wasn't very time consuming, as long as the users know how to solve it.

6.2.2.6 Task 6

This task was to find out how long a specific task has been under the same status. The information was supposed to be found in a table, but that information wasn't given to the participants. The average time to complete the task was 111 seconds, which is very misleading. The actual average time for completing should have been much less, but since the access to Internet worked poorly, it was hard for the participants to scroll in the table. They found the information on the correct place in less than 15 seconds, but since there was a delay, they often scroll passed the information backward and forward, and could not find the information directly like they should have done under different circumstances. Especially the last participant, when it took 245 seconds to find the information, while she was looking at the right place after 15 seconds. All of the participants disclosed at the debriefing that they would have appreciated if the titles of the rows had disappeared when they scrolled in the table. This would have made it easier to find the right information faster. In this case they read the title, then scrolled and then had so scroll back because they forgot which row to look in. This was a bug that was not predicted, since it appeared when using a different browser than the one that was used when developing the tool. It's also worth mention that two of the participants took a shortcut by instead of searching the table manually, they pressed "CTRL + F" to find the correct user story faster, which they succeeded with in both this and the next task. The average number of errors in this task was nine and was divided differently among

the participants. The main reason for the errors in this task was because of the delay of the Internet access. There is again a direct connection between the high average time to complete the task and the number of error. In the debriefing all the participants agreed that there was some frustration of the delay, which caused errors. Only one of the participants had some trouble to find the table. The browser maximized the view of the chart and the table, and therefore the users had to scroll down on the page to find the table. Another thing that doesn't really counts as an error but still worth mention is that was made was that they needed to scroll up to the top of the table again to see which row to look in. This "error" was not counted as an error in the table.

6.2.2.7 Task 7

This task was about searching in the same table as in the last task, but counting the number of tasks belonging to a specific user story. The average time for completing this task was 84 seconds. In this task the first and the last participant had some trouble with accessing the Internet again. All of the participants found the place to find the information directly, but since two of them had problem with a delay, the result is yet again misleading. If the first and the last participant did not have problem with the delay, the average time would have been 20 seconds. One of the problems, besides of the delay, was that four of the five test participants included the actual user story to the number of tasks. And as the last task, it was also hard for them to know what row to look for the correct information since the header of the table disappeared when they started to scroll.
The average number of errors made was one, where only two of the participants made them. The other three participants completed the task correctly without making any errors. One of the errors, that both of the participants made, was that they included the actual user story as a number of task. Another error made by both participants is that they didn't realise that there was a difference between "user story" and "task".

6.2.3 The SUS questionnaire for the hi-fi prototype

The average result of the usability test was 60.5 points. According to SUS, a score under 68 point in considered less usable (see 3.3 System Usability Scale). However, most of the participants wrote on the questionnaire that the main reason that they gave the tool at low score was because of the Internet connection worked so poorly. Everyone agreed on this during the debriefing as well. If the system would have responded faster than it would have been easier to use and wouldn't cause any frustration. Since the internet access is an external feature that affects the system but is hard to predict, it's hard to estimate how the score would have been if the internet access would have been working better. Another thing the participants complained about was that they had a hard time to understand the task at first since they haven't worked in a similar tool before. But after reading the task more thoroughly, they thought it wasn't very hard to solve the task. If the description on the chart was given at the start of the task instead of given it at the end, then the participants would more likely seen it. An example on this was on one task when it said at the end that

the participants should stay on the same page after the task was completed but one of them actually read and stayed on the page. All of the other missed the text and closed the page before moving on to the next task. There was also a pattern of the one that has worse internet access also made more mistakes, took longer time to complete the tasks and then also answered more negatively then the ones that had a better internet access.

6.2.4 Summary of the final usability tests

The final usability test didn't go as good planed, much because of the characteristics of the test participants and the lack of Internet access. If the test would be conducted again, the test participants' characteristics would be Axis employees that work with agile software development and is familiar with how issues are handled currently at Tools, but not familiar of the prototype this master thesis held. This is a target group that is quite hard to find and even if it would be possible to find enough test participants to conduct a second test, preparing and conducting the test would be too time consuming for this master thesis.
The result of the usability test would probably be higher if the Internet access would be better. It test could be conducted again on other students. If the usability tests was conducted on other student that still didn't have the right competence of working with agile software development at Axis, the test results would still be considered as low. Therefore, the usability tests would still not be useful to conduct a second time.
If the lo-fi prototype would be evaluation on students and the usability tests conducted on Axis employees the outcome would be different. The result of the lo-fi prototype would probably be much lower than the result that was actual given because the students would still not see the usefulness of the tool since they still don't have the experience as needed.

6.3 Design

The design of the tool was considered well adjusted for the purpose based on the questionnaire. The test participants thought that there was a good overview of the system and that there was a natural mapping throughout the tools. All charts were design similar so that there was a consistency in the headlines and tools. All users thought it was easy to navigate through the system and the relevant information was accessible in the right places.
Some of the participants asked about the selection of the colours. For example is was considered strange that the status "In process" was in the colour red. This was something that is hard to decide since the users of the tool can name the statuses to anything they like. They can also use as many statuses as they like, so therefore the colours that are chosen for each status is random. However, all statuses in the same project are given the same colour and all of the colours are unique so that the statues are clearly separated and consistent.
All functions that were visual were implemented, which made the tool easier to test compared to if they were simulated.
Donald Norman described in the three conceptual model how a system is be design and implemented on the basis of the designer's mental model. The users

of the system then develop a mental model of how they think the system works through interaction of the system they use. This is something that the developer of the system needs to use for anticipating the user behaviour when using the system and the response when using it. This was something that was considered during the design phase of this master thesis. By making the design of the entire system coherent, the users mental model of how they anticipate the system should response becomes clearer and more intuitive. To make the interaction more frictionless, all buttons and other functions are consistent even with the same scale of colour. For e.g. all buttons for choosing date exactly the same so that the user knew what respond from the system to be expected.
Another thing that makes the interaction with the system usable is to make useful information visible for the user. Information that is used the most should be easiest to find and less used information should be less visible. The users of this system is most likely users that have been using similar tools and charts in different systems. Therefore, an explanation of each chart is something that should be available, but not shown by default. This information was inserted in a button marked by an icon that was shaped as a question mark. By making this choice, a user that needs this information can easily find the right information, but users that doesn't need the information can ignore it. If the information was visible at all time, then the overview of the system would have been perceived as cluttered and the user would have to search for the right information unnecessarily. The information that is useful is placed in the middle of the webpage and the size of each function if based on the importance and frequency of use.

6.4 Comparison

Using HighChart made it easier to customize the design of the charts. It was convenient that both Bootstrap and Javascript was easy to combine with HighChart. Using alternative charts would probably also work just fine, but it would take more time to learn.
Developing in Python facilitate much of the result. Python provides loads of thoughtful packages that demand less of the developer. The learning curve is quite high compared to some other languages, but the time was probably recouped considered all functions that were given.

6.5 Alternative debriefing sessions

6.5.1 "Devil's Advocate" Technique

There are loads of debriefing techniques that serve various purposes. The Devil's Advocate technique is used for situations where you want to retrieve the participant's true feeling regarding the system. In many situations the participants feel reluctant to criticise the system. The test moderator of the debriefing session could for example act surprised when the participant is claiming something positive about the system or asking the participant to explain the reason why something is positive. In some cases the test moderator could even lie and mention that other test participants has answered negative about the same questions to make the current participant feel affinity. (Rubin, J. and Chisnell, D., 2008).

By using this technique, there can be somewhat more confidence to the results. However, this technique could lead participants who are overly ingratiating to change their answer depending on what the test moderator may lead them to. Therefore it may be better to just be natural when asking the questions and hope that the participants will answer as honestly as possible.

7 Conclusion

The conclusion of the first evaluation was that the lo-fi prototype was considered very useful for project managers and project leaders. The usability was high and the functions were relevant for this type of projects. By adding the features, the efficiency would increase since the prototype would make it easier to predict bottlenecks and make accurate forecasts. There was a request for more functions that were more custom made for specific projects.

The final usability test was performed on students. The score for usability from this tests was lower than the first tests. The main reason for this was because of the lack of Internet access that made the tool less useful. This made the tool harder to use since the feedback was delayed and caused frustration. Another reason for the lower score was because the participants previous knowledge of this kind of tool. It was hard for them to see the purpose for some of the functions. The same questionnaire was used for both test groups, which was a mistake. The seconds questionnaire should have been designed more general so that the student could relate more to the tool and the questions.

The summary of the conclusion was that the tool in considered with a good usability, but was customized for users that have been using similar tools and is used to working with R&D projects and is therefore familiar to working with Kanban and Scrum.

8 Bibliography

Agile Alliance (2013). "What is Agile Software Development?". Visited on 23/02/16 Anderson and Reinertsen (2004). "Kanban". https://www.crisp.se/gratis-material-och-guider/kanban. (Visited on 03/10/15)

Atlassian (2016). https://www.atlassian.com/git/ (Visited on 01/11/15)

Berndt and Jönsson (2011) "Agile project management - The software development processes" http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId =1981945&fileOId=1981950. (Visited 23/02/2016)

Bootstrap- Getbootstrap (2016). http://getbootstrap.com/ (Visited on 07/11/15)

Brooke (1986). "SUS -A quick and dirty usability scale". https://books.google.se/books?hl=sv&lr=&id=IfUsRmzAqvEC&oi=fnd&pg =PA189&dq=System+Usability+Scale&ots=G9mzGflk6i&sig=ELdKUuJXKs56N3v9 J_PXC1zYuVc&redir_esc=y#v=onepage&q&f=false (Visited on 05/10/15)

Cederholm (2006). "Bulletproof web design"
Django (2016). https://www.djangoproject.com/ (Visited on 01/11/15)

Fusioncharts (2016). http://www.fusioncharts.com/javascript-charting-comparison/ (Visited on 11/12/15)

Gerrit (2016). https://www.gerritcodereview.com/ (Visited on 01/11/15)

Google charts (2016). https://developers.google.com/chart/ (Visited on 03/10/15)

HighCharts (2016). http://www.highcharts.com/ (Visited on 08/11/15)

Hollerup, Hermansson and Green, (2015) "Agila mjukvaruprojekts största utmaning" http://lup.lub.lu.se/luur/download?func=downloadFile& recordOId=5474157& fileOId=5474158. (Visited on 29/09/15)

Kniberg and Skarin (2013). "Kanban och Scrum –få det bästa av två världar". http://www.infoq.com/resource/news/2010/01/kanban-scrum-minibook/en/resources/KanbanAndScrum-Swedish.pdf (Visited on 29/09/15)

Jönsson (2013) "Agile Development and User-Centered Design - a case study at Sony Mobile Communications AB" http://lup.lub.lu.se/luur/download?func =downloadFile&recordOId=3802849&fileOId=3802862. (Visited on 23/02/2016)
Norman (1990). "Mental models".

Norman (1990). "The design of everyday things".

Python (2016). https://www.python.org/ (Visited on 01/11/15)

Rogers, Sharp and Preece (2011). "Interaction design – beyond human-computer

interaction 3rd edition".

Rubin and Chisnell (2008) "Handbook of Usability Testing (Second Edition) - How to Plan, Design, and Conduct Effective Tests. http://ccftp.scu.edu.cn:8090/Download/efa2417b-08ba-438a-b814-92db3dde0eb6.pdf. (Visited on 29/09/15)

Taiga (2016). https://taiga.io/ (Visited on 01/11/15)

X3schools "HTML". http://www.w3schools.com/HTML/ (Visited on 05/11/15)

X3schools "JS". http://www.w3schools.com/js/ (Visited on 05/11/15)

X3schools "CSS". http://www.w3schools.com/css/ (Visited on 05/11/15)

X3schools "AJAX". http://www.w3schools.com/ajax/ (Visited on 05/11/15)

YUI Charts, 2016. http://yuilibrary.com/yui/docs/charts/ (Visited on 11/12/15)

Appendix

Test plan - Streamlining the handling of R&D projects

Purpose, goal and objectives of the test.

The goal of the usability tests is to understand if the target audience of the tool can understand and use it and how usable it is. Research questions. The research question will describe the questions that need to be resolved by conducting the usability test.

1. How easily do users understand how to read the charts and tables?
2. How easily do users understand how to map the charts to the tables?
3. How closely does the flow of the software reflect how the user thinks of the work flow?
4. How well do they understand the content of the topics they find?
5. Which parts of each topic do users pay attention to?

Participant's characteristics.

The number of participants for the usability test will be five to ten persons. Student that are aspiring project managers is the target audience for the usability test. This audience are in the age range of 25-35 and are well familiar with the terms as Kanban, usability and affordance. Most of the students have not yet worked with similar tools, but are familiar with the theoretical background.

Task list.

1. The test moderator must answer a feasibility study on the internet before the physical test.
2. Test moderator presents the test subject to test.
3. The test person must sign a legal document saying that it is alright to test the person will be filmed and the data will be used later in the project is completed.
4. The test person will receive an instruction of the test. Filming of the test begins.
5. The test person reads the first task.
6. The test person performs the first task. Any question and answer session with the test person containing subjective questions.
7. The test person reads the next task.
8. The test subject performs next task.
9. Number 7-8 will be repeated until all given tasks are finished.

10. After the test is finished, the test person must answer a questionnaire.

Test environment, equipment and logistics.

All test will be performed in an isolated test environment. The room will be set up with cameras and all equipment's to document the entire session.

- Cameras and microphones that is included in the test lab.
- A computer to conduct the test on.
- Two chairs and a table to perform the test on.
- The test will be performed on a remote computer places on Axis.
- The computer on Axis will be started with the program started.
- Nomachine installed in the computer to access the computer on Axis.
- A pen and paper to register all observations.
- The test plan.
- All scripts for conducting the tests.

Test moderate role.

Since this is a single-person master thesis, all roles will be assigned to one person. The test moderator will sit in the monitor room with the test participant during the entire test session. The test moderator will first introduce the session, and then perform a short background interview, which will be read from a script. After that all task will be introduced subsequently from a script. The test moderator may ask unscripted follow-up questions if needed to clarify the test person's behaviour, comments and expectations. During the entire session the test moderator will take detailed notes and records.

Data to be collected and evaluation measures.

The questions will be answered with different types of data collected. The questions:

1. How easily do users understand how to read the charts and tables?
2. How easily do users understand how to map the charts to the tables?
3. How closely does the flow of the software reflect how the user thinks of the work flow?
4. How well do they understand the content of the topics they find?
5. Which parts of each topic do users pay attention to?

The data to be collected to each question:

- Number and percentage of tasks completed correctly with and without prompts or assistance.
- Time to complete each task.

During the test, performance data will be collected such as measures of behaviour like number of error rates, time, and counts of observed behaviour elements. Since there will be a recording of the tests, performance data can be complemented afterwards.

After the test preference data will be collected, which consist more of a subjective data. This data will be gathered from the debriefing session by rating scales that measure the participant's feelings and/or opinion of the system.

Test assignments

Scenario 1:
Du ska undersöka hur många User stories som finns under kategorin "Done" den 5 januari 2016.
Uppgiftsbeskrivning:
Gå till status count-menyn för projekt 239. Ta reda på hur många User stories som finns under kategorin "Done" datumet den 5 januari 2016. När du är klart kan du gå tillbaka till huvudmenyn.
Korrekt utförd när: Testpersonen har svarat på det antalet User stories som personen tror finns under kategorin "Done".
Tidsåtgång: 3 minuter.
Maximal tidsåtgång: 5 min, sen avbryts uppgiften.
.....
Scenario 2:
Du ska undersöka hur många User stories som finns under kategorin "New" den 1 januari 2016.
Uppgiftsbeskrivning:
Gå till All stats-menyn för projekt 239. Ta reda på hur många User stories som finns under kategorin "New" datumet den 1 januari 2016. När du är klart kan du gå tillbaka till huvudmenyn.

Korrekt utförd när: Testpersonen har svarat på det antalet User stories som personen tror finns under kategorin "New".
Tidsåtgång: 3 minuter.
Maximal tidsåtgång: 5 min, sen avbryts uppgiften.
.....
Scenario 3:
Du ska nu göra en forecast baserat på hur det gått för dig och ditt team mellan den 1 november 2015 till den 1 januari 2016. Forecasten ska vara medianen av ditt uppskattade värde.
Uppgiftsbeskrivning:
Gå till Cumulative flow chart-menyn för projekt 239. Gör en uppskattning på hur flödet kommer se ut om 1 månad om man baserar flödet mellan den 1 november 2015 till den 1 januari 2016. När du är klart kan du gå tillbaka till huvudmenyn.

Korrekt utförd när: Testpersonen har ställt in ett värde på startdatum och ett värde på slutdatum eller testpersonen anser att uppgiften är klar.
Tidsåtgång: 5 minuter.
Maximal tidsåtgång: 10 min, sen avbryts uppgiften.

.....

Scenario 4:
Du ska nu skapa en Burndown chart för dig och ditt team. Burndown ska vara mellan den 20 december 2016 till den 20 februari 2016.
Uppgiftsbeskrivning:
Gå till Burndown-menyn för projekt 239. Din Burndown ska starta den 20 december 2015 och sluta den 20 februari 2016. Efter uppgiften är klar kan du stanna kvar på sidan.
Korrekt utförd när: Testpersonen har ställt in ett värde på startdatum och ett värde på slutdatum eller testpersonen anser att uppgiften är klar.
Tidsåtgång: 5 minuter.
Maximal tidsåtgång: 10 min, sen avbryts uppgiften.

.....

Scenario 5:
Du ser att du antaglien inte kommer hinna lösa alla task inom tidsramen i förra uppgiften. Du ska nu jämföra rate mellan två olika burndowns för att se vilken rate som behövs för att klara tidsramen. Uppgiftsbeskrivning: Låt den burndown-uppskattningen från den förra uppgiften vara kvar. Ställ in slutdatum efter vart burndown-uppskattningen kommer att klaras och läs av vilken rate som krävs för att klara dead-line*. *OBS! För att deadline krävs att hela stapeln är under burndown-uppskattningen. Du kan gå tillbaka till huvudmenyn.
Korrekt utförd när: Testpersonen har ställt in ett värde på startdatum och ett värde på slutdatum på compare-burndown eller testpersonen anser att uppgiften är klar.
Tidsåtgång: 5 minuter.
Maximal tidsåtgång: 10 min, sen avbryts uppgiften.

.....

Scenario 6: Du tycker att det går väldigt långsamt i projektet men har svårt att uppskatta hur lång tid det tagit. Du vill därför kolla hur länge User storien “Granska kod” var varit i samma kategori.
Uppgiftsbeskrivning:
Gå till All stats-menys för projekt 239. Kolla i tabellen hur länge User storien “Granska kod” har varit i samma state. Efter uppgiften är klar kan du stanna kvar på sidan.
Korrekt utförd när: Testpersonen har gått in på menyn “All stats” och kollat i tabellen efter rätt User story.
Tidsåtgång: 3 minuter.
Maximal tidsåtgång: 5 min, sen avbryts uppgiften.

.....

Scenario 7:
Du vill veta hur många task User storien “Genomgång av Alexs kod” har.
Uppgiftsbeskrivning:
Leta upp User storien “Genomgång av Alexs kod” har och räkna antalet tasks som den User storien har.

Korrekt utförd när: Testpersonen har gått in på rätt meny och räknat task tillhörande User storien i tabellen.
Tidsåtgång: 5 minuter.
Maximal tidsåtgång: 10 min, sen avbryts uppgiften.
Post-test questionnaire
The participants will receive a questionnaire directly after the usability test in the same room as the test was.
Approximate time estimate: 5 min
.....
Debriefing session
The debriefing session will not be so formal since it will be based on the test results and post-questionnaire.
Approximate time estimate: 10 min