Designing a Project Management Application for Agile Software Development

Ellinor Rönn

Ellinor Rönn
HT 2016 · VT 2017
Master thesis, 30 hp
Supervisor: Håkan Gulliksson
Extern Supervisor: Idamaria Harnesk
Examiner: Thomas Mejtoft
Master of Engineering in Interaction Technology and Design
Abstract

Agile software development has emerged as a response to the rapidly changing software environment and the need to anticipate late change of requirements. An important tool for many agile methods is a project management application with a task board to keep track of the progress. However, it can be difficult to find a project management application that is suited for a workplace where development teams use different methods.

The aim of this master thesis was to develop a prototype of a project management application adapted to Codemill AB where development teams use different agile methods. To achieve this, the work process of development teams were investigated, as well as existing project management applications, and the most important functionality and features from a user available point of view were identified. Two design concepts were developed based on the users’ needs. Both concepts were tested and compared by end users in order to develop a refined prototype with the best parts from both concepts.

The results indicate that the development process in a workplace can vary a lot between different development teams even though their methods are inspired by the same framework. A project management application for agile software development must be flexible enough to handle these differences without becoming too complex and difficult to use. None of the investigated applications achieved this balance. This work aimed to develop a prototype of an application that fills this gap, with the users’ needs in focus. The findings in this thesis are based on Codemill’s needs, however, they could also be considered by others with similar needs.
Acknowledgements

I want to thank Codemill and all employees there for giving me this opportunity and for participating in surveys, workshops and tests. Extra thanks to my supervisor Idamaria Harnesk and team Machine, You have been of great help and it has been a pleasure working with you. Furthermore, I want to thank Håkan Gulliksson for the help with the report. Finally, I would like to thank Christofer Ryås for all his support during this work.
Contents

1 Introduction 3
   1.1 Aim and objective 3
   1.2 Codemill 4
   1.3 Overview of the work 4

2 Agile software development 5
   2.1 Scrum 6
      2.1.1 Scrum Events 7
   2.2 Kanban 7

3 Development teams 9
   3.1 Method 9
   3.2 Result 10
      3.2.1 Sprints and iterations 11
      3.2.2 Task board 11
      3.2.3 Project management applications 13

4 Requirement analysis 15
   4.1 Method 15
      4.1.1 Questionnaire 15
      4.1.2 User story workshop 16
   4.2 Result 17
      4.2.1 Questionnaire 17
      4.2.2 User story workshop 19

5 Competitor analysis 21
   5.1 Method 21
   5.2 Result 22
      5.2.1 Trello 22
5.2.2 Pivotal Tracker 25
5.2.3 ScrumDesk 30

6 Prototype

6.1 Method
6.1.1 Design concepts
6.1.2 Usability tests
6.1.3 Refinements
6.2 Result
6.2.1 Design concepts
6.2.2 Usability tests
6.2.3 Refinements

7 Discussion
7.1 Limitations
7.2 Future work

8 Conclusions

References

A Requirements
A.1 Requirements
A.1.1 User stories
A.1.2 Questionnaire result

B Comparison test
B.1 Product backlog, concept 1
B.2 Product backlog, concept 2
B.3 Agile board, concept 1
B.4 Agile board, concept 2

C Sketch workshop
C.1 Adding items in story map
C.2 Sprint planning in Product backlog
1 Introduction

Traditional software development methods, such as the waterfall model and the spiral model, have widely been used for more than 40 years [1]. They support well defined planning and assume a large amount of documentation [1–3].

It is common with a rapidly changing business environment, which introduces challenges to traditional development methods. Non-changing requirements produced in the beginning of the project may be outdated before the project is completed [4]. The changing business environments and the need to anticipate late change of requirements required new approaches, which are now called agile methods [3]. The aim of agile methods is to answer the need to develop software quickly in an environment where the requirements are rapidly changing [5, 6].

A central part of several agile methods is user stories, which are short descriptions of parts of the work a user does with the system, formulated in everyday or business language of the end-user [7]. User stories are often used in requirement engineering, release and iteration planning, and for tracking the progress of a project [4]. When tracking the progress of a project, the task board is an important tool to the agile team [8]. A task board is a chart of columns that is used to track the state of the tasks done in a project. One early method for this was to use an Excel spreadsheet [8]. Another early method was to write the stories on index cards which can be displayed and arranged on a pin board. Today there are a great number of software tools for this, providing different sets of features [4].

However, it can be difficult to find a project management application that is suited for a workplace where the development teams use different methods. If you start looking at different applications you may soon notice that many of them are either too simple for agile software development or they have so many possibilities that they become too complex and difficult to use. Other applications only allow one specific method that you have to follow strictly. This may, much like the too simple applications, force the development team to adapt their process after the tool, which should be avoided [9].

One company that has encountered this problem is Codemill AB, where there are currently five development teams, using different methods. The teams are mainly using Trello [10] to keep track of their projects since it is the tool they have found that best meet their needs. However, they are not satisfied with any tool they have tried since they are not adapted to how they want to work at Codemill.

1.1 Aim and objective

The aim of this Master Thesis is to develop a prototype of a project management application adapted for Codemill, a company using agile software development, with focus on the users need. The following objectives were formulated:
• Investigate and report on how development teams are organized and work at Codemill.

• Identify the most important functions in a project management application from the users’ point of view.

• Investigate existing project management applications. Examine how they work and what their strengths and weaknesses are.

1.2 Codemill

Codemill is an IT company founded 2007 in Umeå, Sweden. Mainly they help companies in the fashion and media industries by developing software to maintain, analyse and publish video online. At Codemill there are currently five developing teams that work, according to their process document, either by Scrum or Kanban. In reality the teams use their own methods inspired by Scrum or Kanban rather than follow them strictly since methods that does not follow every rule of the framework is here referred to as "Scrum inspired" or "Kanban inspired".

1.3 Overview of the work

User-centered design was used to develop the prototype. A central part of user-centered design (UCD) is the focus on the users’ needs, wants and abilities [11]. Other UCD characteristics are that users are involved in the design process early and often, and design solutions are iterated [11][12]. The users were involved from the beginning, by surveys and a workshop to identify requirements for the prototype. Existing systems were analysed to find good solutions and what to avoid. Two design concepts were developed and tested on end users to analyse their strengths and weaknesses and examine which approach was best. Based on the user tests, a refined prototype was implemented with the best parts from both concepts.
2 Agile software development

In 2001, 17 software experts participated in a meeting where the Agile Software Development Alliance was founded [1,13]. The purpose of their meeting was to uncover better ways of developing software. This resulted in a Manifesto for Agile Software Development, signed by all participants [13]. Two participants were Fowler and Highsmith [13], who describe four values of which agility is based on:

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

The main purpose of agile methods is to enable fast development of software in an environment where the requirements are constantly changing [6]. Agile methods focus on iterative and incremental development, frequent delivery, collaboration between customer and developers, continuous quality improvement, simplicity and attention to customer satisfaction [1,13,14]. The teams should be self-organizing, maintain a constant pace and regularly reflect on how to become more effective and adjusts accordingly [13].

A central part of several agile methods is the use of user stories, which are often used in requirement engineering, release and iteration planning, and for tracking the progress of a project [4,7]. A user story is a short description in the user’s own words of what they want to do with the system. This enables developers to communicate with customers and end users which functionalities are to be implemented, without the need of familiarity with a specific method or jargon [4,7].

Breaking down the requirements into smaller chunks, stories, enables visibility of the development progress. Building the product story by story, with full integration to the system, enables everyone to see the product grow. However, with this chunking it is easy to loose the big picture of what the system should do, which may cause a product that is not helpful for the users [15]. Fowler suggests the technique story mapping to provide the big picture that easily can be missed with a pile of stories [15]. When the features of the system have been gathered as user stories, describing what the user can do with the system, the user stories should be ordered in sequential order horizontally. The user stories should then be ordered vertically after how critical they are [16].

Before the development team starts implementing the stories, the stories are often broken down into smaller tasks that have to be done to complete the story. The task board is an important tool used by an agile team to track their progress with cards representing these tasks [8]. A task board has vertical columns that represent the state of the tasks within.
A simple example of a task board is shown in Figure 2.0.1, where each story is presented in its own swimlane together with its tasks. The column Sprint backlog contains all stories to do during the current iteration, Todo contains all not started tasks of the story, In progress contains all started tasks of the story and Done contains all finished tasks of the story. The tasks should flow from the left to the right during the development work and when all tasks for a story are in the Done column, the story can be considered complete and ready for review and approval of the product owner or customer [8]. The example in Figure 2.0.1 only presents the title on the cards. However, it is common that a development team presents more information on these cards. A story card could for example have ID, name, importance, estimate/size, how to demo, notes, track/category, components in for of check boxes, requestor (which stakeholder asked for the feature), bug tracking ID and assignees [8,9].

### 2.1 Scrum

Scrum is defined by the founders as "A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value" [17]. The framework consists of a Scrum Team with associated roles, events, artefacts and rules that bind everything together [17][18]. The team consists of a product owner, a development team, and a Scrum master, each with different responsibilities. The product owner is one person that is responsible for the work of the team and managing the product backlog. The development team is responsible for delivering a po-
tentially releasable increment at the end of every sprint. The development team should, like the Scrum team, be self-organizing and cross-functional, which means that the team decides how to accomplish their work and it has all competencies needed to accomplish the work. The Scrum master is responsible of ensuring that Scrum is understood and used by the team and ensuring external people understand which of their interaction is useful for the Scrum team.

2.1.1 Scrum Events

Scrum uses specified events with a determined maximum duration to minimize the need for meetings not defined in Scrum [17]. One such event is a sprint, which is a predefined period of maximum one month, in which a product increment is created. Immediately when a sprint is done the next sprint begins. A sprint consists of all the other events, which are described by Schwaber and Sutherland [17] as:

**Sprint Planning:** Each sprint starts with a sprint planning where the entire Scrum team collaborates to plan the work to perform during the sprint. Sprint planning should provide answers to which functionality can be developed during the sprint and how this is achieved.

**Daily Scrums:** The development team has a daily meeting of maximum 15 minutes where the team members inspect the work since last daily Scrum, synchronize activities and create a plan for the next 24 hours.

**Development Work:** Implementation of the tasks chosen for the sprint.

**Sprint Review:** At the end of each sprint a sprint review is held with the purpose to inspect the increment and, if necessary, adapt the product backlog. The product owner explains which product backlog items that have been done and which have not. The development team discusses what went well with the work during the sprint and which problems occurred.

**Sprint Retrospective:** Each sprint ends with a sprint retrospective where the team discusses what went well during the sprint and what should be improved for the next sprint. The team inspects itself with regards to people, relationships, process and tools and creates a plan for improvements to implement in the next sprint.

2.2 Kanban

Kanban is more adaptive than Scrum, which means that there are fewer rules to follow. When using Kanban, there are only two rules, to visualize the work flow and to limit the work in progress [14]. In order to visualize the work flow, the work is broken down into small tasks, which are written on cards and put on a wall, or a task board, explained in the beginning of chapter 2. This task board can be either physical or digital.

The work in progress is limited by a maximum limitation of tasks in each column, which is usually shown together with the column name on the task board [14]. The purpose of the work in progress limitation is to find bottlenecks early and to solve them instead of piling
up unfinished work. The limit should be high enough to keep all team members busy but low enough to react early to bottlenecks.
3 Development teams

It is important to support how users actually work, rather than forcing them to change the way they do something [12]. Therefore, interviews were conducted with users in an early stage to investigate how the development teams work. Rubin and Chisnell state that one attribute of usability is usefulness, which means that the product has to enable the user to achieve their goals and that the users are willing to use it [12]. Furthermore, they emphasize the importance of early investigation of which features are desirable and necessary, before other usability aspects are considered. In order to design a useful project management application, the development teams’ needs and goals of such application had to be investigated.

3.1 Method

Interviews were conducted with the team leaders with the goals to understand how the teams are organized and work, understand the development process, and gather material to use as a foundation for the requirement analysis. Before interviewing the team leaders, a pilot interview was conducted, which is necessary in order to find design faults and to enable an evaluation [19].

The interviews were held in a small conference room with a whiteboard and a TV-screen to connect devices to, enabling visual aids. The respondents were given a short introduction to the project and the purpose of the interview. They were then asked if they allowed audio recording of the interview, that would be used to avoid misinterpretations. Semi-structured interviews can benefit from recorders since it is good to have a complete record of the interview, and the interviewer can focus on the interview rather than taking notes [20].

Face-to-face interviews with semi-structured questions were used to allow the interviewer to ask questions that are not in the script. The purpose of semi-structured interviews is to gather information about some predefined topics while allowing further exploration when new topics or issues appear [20]. Semi-structured interviews are well suited when gathering information about user needs and the development process, such as task flow, work artifacts, equipment, etcetera [20]. It is therefore a suitable technique based on the purpose of the interviews.

Questions and probes were developed and organised after topics in an interview guide. The interview guide contained questions about which agile method the team use and how they use it, different events included in their process, how they use the task board, how they plan the work, which project management application they use and how they feel about it, and what they would want in a new project management application.

When designing survey questions, there are some important things to consider to obtain effective responses that meet the goal of the survey. Stone has constructed some guidelines for
creating good questions that enable the respondent to answer effectively [19]. These guidelines were taken in consideration when developing the interview questions. According to the guidelines the questions should be appropriate, which means that the questionnaire should be capable of providing answers to the questions being asked. This means for example that the interviewer should ask questions in the respondents field of knowledge. This was taken into account by asking about the respondents about their own method and tools rather than asking about a specific method or tool.

Another guideline is that the questions should be unbiased [19]. Open questions were used because they prevent the biased responses that are only reported because of a given alternative, which might be caused by a closed question by forcing a choice between given alternatives. Open questions are suitable when the range of answers is not known beforehand, and to determine which response options should be used in closed questions for a following larger survey [21]. Another reason for using open questions is that they encourage the respondent to answer freely with their own words, which is useful when collecting qualitative data and to investigate respondents’ feelings and opinions [19,22]. Another way to limit the bias was by the use of neutral formulations, such as "describe how..." and "how do you feel about...", [20].

Wilson suggests a duration of half an hour to two hours for each interview [20]. He argues that longer interviews may reduce the number of qualified participants who do not want to lose valuable work time. Furthermore, he states that shorter interviews may not be enough to cover all important topics in sufficient depth. The interviews were planned to take approximately 40 minutes in order to gather all information needed without requiring more time than the participants are willing to give up. The pilot interview took approximately 35 minutes.

### 3.2 Result

Five interviews were conducted, one with each team leader. None of the teams usually follows any agile method strictly. They have created their own method inspired by Scrum and/or Kanban. None of the teams uses the Kanban requirement maximum work-in-progress activities. One of the major similarities between the teams is that they want a close collaboration with the client. It is important that the client is involved in prioritising what is to be done during the time to come, to ensure that the most important things are done when the client stops the project. Furthermore, all the teams visualize their work with a task board where the tasks are represented by cards that is moved between columns indicating their state, for example "Todo", "In progress", and "Done".

Team C always uses the same method while the other four teams use different methods in different projects. In some projects the team has to follow the same method as their client. Team A describes their usual method as Kanban like with Scrum elements such as, daily stand ups, retrospectives and Scrum roles. In many cases when the team starts working with a new project they are given a list of requirements by the client. Other times the team performs the preparatory work and creates a list of stories or epics that are broken down into tasks as Trello cards that clearly explain what should be done. These Trello cards are prioritized by the clients.

Team B usually uses a combination of Scrum and Kanban since it is often difficult to follow
a specific method strictly when the team members are working in different projects. However, all team members will soon be working in the same project and the team will start using Scrum.

Team C always uses a Kanban inspired method where the product owner chooses tasks to do from an idea list and moves them to the Todo column on the task board. When the product owner stops adding cards to the Todo column, a test period starts, during which the product owner tests the system while the team fixes bugs. After this period the project is completed and the client will be offered a support agreement for further assistance. Support clients have a weekly pot for support issues and any time not used during the week is expired.

Team D uses a Scrum inspired method when it is possible, otherwise they use a more Kanban inspired method. Sometimes they work on multiple projects at the same time, which can make it difficult to follow one method strictly. In some cases they pick cards from multiple backlogs and the members pick cards from the project they are most suitable for. The Scrum master may assign team members to different projects, which gives the project with the most assignees the highest priority. In larger projects they use a more Scrum like method, it is then important that the client is engaged in planning, story writing, prioritising and demonstrations.

Team E usually uses a combination of Scrum and Kanban. The team does not plan much what to do during the upcoming sprint. The team members pick cards from the backlog and do as much work as they can. At the end of each sprint, the team meets with the product owner and demonstrates the tasks implemented during the sprint. The respondent says that their work process is based on limitations caused by the project management application they use.

### 3.2.1 Sprints and iterations

All teams work with iterations of some kind. Team A uses periods, with undefined length, which ends with delivery to the client. Every week starts with weekly planning where the team discuss which tasks to do during the week and moves these tasks from the product backlog to a column on the task board.

Two teams (team B and D) use sprints. However, team D does not have a sprint planning. Team C uses weekly iterations, which starts with looking at the tasks for the week and decide who will do what. Once a week the team has a peer review where the whole team goes through the cards implemented during the week to ensure that each card has been understood and implemented. The client sees the same task board and approves an implemented task by moving the card to the column Done.

Team E uses releases, similar to sprints. The team has sprint planning together with the client, where they go through the top of the product backlog, using story mapping and decide which stories to do during the iteration. Sometimes they decide a maximum number of story points to add to the sprint.

### 3.2.2 Task board

Regardless of the method, all teams visualize their work with a task board where the tasks are represented by cards that are moved between columns representing the state of the tasks. All teams use the columns described in the list below. The column name differs between
the teams but the purpose is the same.

- **Product backlog** is a list of backlog items, such as stories, usually prioritised by the product owner moving the most important backlog items to the top of the list.

- **Todo** contains all the tasks to do during the current iteration or time to come.

- **In progress** contains all tasks that someone is currently working on.

- **Peer review** is a list of tasks that have been done and need to be reviewed by a team member.

- **Done** contains all tasks that are done and have been approved.

Four out of five teams (Team A, B, D and E) have meetings where they decide which backlog items to solve during the time to come. Stories are usually broken down into smaller tasks that have to be completed to achieve the goal of the story. These tasks are put in a sprint backlog or a todo list. In team C, the product owner decides which tasks to do by regularly moving tasks from the backlog to the todo list. This team has weekly meetings to go through the tasks to do and decides who will do what.

Two of the teams (team B and D) use a template for how a story should be written. It should have:

- title
- description in form of "As a ... I want ... so that..."
- dependencies - what has to be solved before the story can be worked on
- acceptance criteria - what has be be done manually in order to know that the story is solved
- tasks - what has to be done to solve the story

Team D also uses estimate with interval, e.g. 5-20 days and, when the project management application allows it, and ID to refer to. This team also has rules for stories in the Todo list. It should meet the "ready to work on" criteria, be written in the form "As a ... I want ... so that ..." and any external dependencies have to be resolved. Team A, C and E use a describing text of what to do without any specific template for the stories.

Task cards are not as structured as the stories. They contain information about how ho solve the problem, for example bullet list that describes what to implement, which framework and which file to edit. For support cards it is usually important to know which browser was used, OS version and how to reproduce the problem. For both stories and tasks, it is also common to use check lists with "Todos", attachments (for example sketches), and assign cards to members.

In all teams, when working with the tasks, a team member picks a highly prioritized task from Todo and assigns the task to themselves. The team member moves the task to In progress and performs the task, whereafter the task is moved to the Review list. Another team member ensures that the task is done correctly, and moves the task to Completed. The
flow is however more complex than this with more columns, for example ideas and multiple columns for reviewing a task, and tasks that are not approved at some point it is moved back to the backlog, Todo list or a waiting column with feedback about why it was not approved.

3.2.3 Project management applications

The teams use different tools to visualize the work flow. Four out of five teams (A, B, C and E) use Trello \[10\] most of the time since it is quick and easy to work with. Team D also uses Trello sometimes but most of the times they use Redmine \[23\] since they think it has better support for Scrum and story mapping. This respondent does not like Redmine since it is "old, ugly and difficult to work with". The respondent says that Redmine requires many clicks for important functions, such as getting to the task board. This team leader has to configure the system so that all stories are shown and it is difficult to see when a story is added which makes it easy too loose stories. Furthermore, it is difficult to get the clients to use it since they have to log in every time and it takes many clicks to get to the task board. However, this respondent mentions that the other members of the team likes Redmine when all the configurations are made.

All teams agreed that one of the major advantages with Trello is that it is quick and easy to use. It is easy for anyone to use, which makes it easier to invite clients to prioritise the backlog. Furthermore, drag and drop makes it easy to move the cards. One respondent says that Trello has good shortcuts, plugins, it is easy to add check list and create new cards from items in check lists. Another respondent mentioned that Trello has a good mobile client a third respondent said that it is good that the system is flexible so that anyone can do anything.

However, four out of five respondents think that Trello is too simple and is missing important functionality. For example, they think Trello has lacking support for filtering, sorting and ordering cards. Furthermore, there is only one type of cards, they want to be able to distinguish between stories and tasks. There is also only one view over the project and there is no support for story mapping. Respondents miss ID on the cards so that one can refer to other cards or link to a version control system. One respondent also misses the possibility to change a member’s rights so that a guest/client can log in and see/change the backlog but not being able to make changes in the task board. Two teams say that Trello forces the teams to adapt their process to a too simple tool, which is not good.

The respondents want the following in a project management application:

- The system should be easy to use both for the teams and external clients. This means for example that you should be able to move cards with drag and drop and all cards should be shown as default.

- A good way to create the backlog, for example story mapping. There should also be multiple levels of cards so that one can create sub tasks to stories. A template for the stories could ensure that they are written correctly.

- A card should be able to be assigned to multiple members, have support for estimate, have support for state more than the columns and perhaps rules for how to move the cards to avoid mistakes.

- Good support for bugs and support issues. It should be easy for clients to report bugs
and other support issues. Furthermore, it should be easy to follow the progress of a reported issue. As a developer it is important to get information about how to recreate a bug and in which environment it occurred. It could be good to see on a card what has been done in a bug fix instead of having to use another system.

- Integration with time reporting system, for example start a task and the time reporting system logs the time.

- Connection with version control system, for example link a card to a branch in git.
4 Requirement analysis

The previous chapter investigated which features should be considered in the prototype. This chapter investigates which of these functions are desired by a larger group of users.

4.1 Method

Based on the interview results, different functions and characteristics were compiled and used in a questionnaire with the purpose to rank the most important things to include in the prototype. A questionnaire was used because it makes it possible gather large amounts of data and understand the preferences of a broad base of users [12, 24]. A user story workshop was held to further investigate a few of the most important functions. The results of the questionnaire and the workshop were used to create a requirement specification of the prototype.

4.1.1 Questionnaire

The result from the interviews in [chapter 3] was used to compile possible functionalities and features of a project management application. For example support for sprints, possible ways to manage the product backlog, functionality for the task board, filter and order cards, attributes of different types of cards, roles and rules, etcetera.

A questionnaire was created with the purpose to rate the compiled functions and features. The questionnaire was sent to all employees at Codemill to investigate which of these functionalities and features are desired by a larger group of users. Google Forms was used to create a questionnaire since a web based questionnaire can be answered even by employees not stationed at the office. Furthermore it has support for collecting and analysing results which saves time. According to Stone, all survey should be piloted to find and correct design flaws and to evaluate the responses to ensure that the correct thing is measured [19]. Therefore, the questionnaire was reviewed by an employee at Codemill and then pilot tested by an end user to ensure that it is understood, covers important topics, and have enough response alternatives.

The questionnaire consisted mainly of closed questions, providing the respondent with alternatives such as yes/no, grading scale, checklist or alternative statements [19]. The closed questions were used because questions with given alternatives are easy and quick to answer, compared to open questions. Furthermore, they are easier to analyse and code, which is useful for collecting quantitative data. Closed questions are also more reliable than open questions because of the fixed answer choices.

The response to a closed question could however be biased by forcing a choice between given alternatives. The limitation of the alternatives could prevent any other responses to
Table 1 The table shows how the features were prioritized. For example, features with priority must have the average value of 1 or higher and/or at least 75% of the participants thought it was important or very important.

<table>
<thead>
<tr>
<th>MoSCoW</th>
<th>Avg. value</th>
<th>Important or very important</th>
</tr>
</thead>
<tbody>
<tr>
<td>Must</td>
<td>$\geq 1$</td>
<td>75%</td>
</tr>
<tr>
<td>Should</td>
<td>$\geq 0$</td>
<td>50%</td>
</tr>
<tr>
<td>Could</td>
<td>$\geq -1$</td>
<td>25%</td>
</tr>
<tr>
<td>Won’t</td>
<td>$&lt; -1$</td>
<td>0%</td>
</tr>
</tbody>
</table>

be reported. This bias was limited by adding an “other” category with space for a comment for all questions with alternatives. This is necessary in order to cope with most possible responses [19]. Similarly, for every grading scale question there was an alternative “No opinion/I don’t know”.

The response alternatives with rating scale were valued between -2 to 2, which was not presented in the questionnaire. Questions such as "How important is it that the system supports...?" had the following alternatives with the value presented in brackets:

- Not at all important (-2)
- Somewhat important (-1)
- Important (1)
- Very important (2)
- No opinion/I don’t know (0)

The requirements were prioritized with MoSCoW (Must, Should, Could, Won’t). The rating questions were prioritized based on their mean value and percentage of respondents that thought the feature was important or very important (see Table [1]). MoSCoW for multiple choice questions were determined similarly. However, average value was not used and a positive response is a checked box. Must requires 75% positive responses, Should requires 50% positive responses and Could requires 25% positive responses.

4.1.2 User story workshop

A workshop was held with the purpose to further investigate three topics, namely backlog, sprints and support issues. The goal of the workshop was to create a set of user stories that describe what the user should be able to do with the system. Generating user stories in group allows problems to be seen from multiple perspectives and one person’s ideas may trigger another person’s ideas which may generate more user stories than generating user stories individually [7, 25].

There are however some difficulties that have to be taken in consideration, such as securing collaboration and progression towards satisfactory outcome [25]. A leadership is a good way to make the interaction productive, effective and efficient [25]. Therefore, a workshop leader participated to ensure that the participants collaborated, and kept a positive atmosphere by giving all participants possibility to speak their mind without being criticized.
The workshop leader also made sure that the participants kept focus on the task and decided when to move on to the next task.

In addition to the workshop leader, three end users participated in the workshop. The workshop started with introducing the participants to the work with the prototype, the purpose of the workshop and the procedure of the workshop. The first task was to identify all the possible users of the application. All ideas were discussed and refined. The resulting user roles were written on big index cards that remained on the board during the whole session. These roles were used when creating the user stories.

The leader wrote a topic at the top of the whiteboard and explained it to the participants, who wrote user stories explaining what they would like to do with the system. The user stories were placed on the whiteboard in a natural flow, from left to right. The participants discussed, grouped and refined the user stories. In some cases, epics (larger stories) were created or a user story was broken down into smaller ones. The leader asked if everyone agreed on all the user stories, if necessary, the participants discussed to agree on something.

4.2 Result

Based on the questionnaire and user story workshop, requirements, user stories and user requests were compiled (see Appendix A).

4.2.1 Questionnaire

The questionnaire was answered by 17 employees at Codemill, with different roles and using different methods. The questionnaire resulted in a list of functionality and features ordered after MoSCoW (Must, Should, Could or Won’t). Features with priority ”Must” have to be implemented, priority ”Should” are not critical but have a high value to the users and should thereby be implemented. Features with priority ”Could” could be implemented if they do not require too much effort. Features with priority ”Won’t” will only be implemented if they facilitate implementation of higher prioritized features.

The system must:

- Have a list view over the product backlog
- Provide possibility to sort items after priority and state (should also have functionality to sort items after type/category and release/sprint, and could have functionality to sort items after estimation, creation date, and deadline)
- Provide possibility to filter items after search word, assignees, type/category, state and Release/Sprint (should also have functionality to filter items after priority and could have functionality to filter after requester and creation date.
- Provide possibility to prioritize cards with placement in list (could also have support for MoSCoW)
- Support sprints (that should have goal, date and stop date, and could have possibility to set a maximum limit of story points)
• Have functionality to assign a card to a team member
• Enable the user to move cards with drag and drop

The system should:

• Enable a user to see members in the project
• Indicate that a client has approved the implementation of a card

The system could:

• Have functionality to create story maps
• Have functionality to change view of backlog and task board with regard to the amount of information presented on the cards.
• Provide information about activities on the task board
• Automatically assign a card to the member moving it to the column In progress
• Have functionality to create cards from check lists in other cards
• Provide rules for how the cards can be moved (should have no rules or selectable rules)
• Be integrated with time reporting system

The system won’t:

• Support a maximum limit of tasks in the task board columns
• Provide predefined columns
• Support retrospectives
• Present burndown charts
• Support roles that controls what a user can do with the system

A story must have title, description, possibility to add attachments, possibility to comment on the card. It should also have ID, estimate, creation date, type/category, definition of done, acceptance criteria, dependencies, related tasks and provide information about activities related to the story. Furthermore, a story could provide information about its requester.

A task must have title, description, reference to story, assignees and possibility to comment on the card. It should also have ID, possibility to add attachments and provide information about activities related to the task. Furthermore, it could have estimate, creation date, deadline, type/category, link to version control system. A bug card must have additional information about how the problem can be reproduced, and contact information to a person the developers can contact for more information. A bug card should also provide information about the environment in which the problem occurred, such as operating system, browser, server, etcetera.
The cards must provide functionality to move the card to a specific place, remove the card, copy link to card, and subscribe to the card to get notifications about activities related to the card. It could also provide functionality to make a copy of the card. A task card should provide functionality to be converted into a story and the system could provide functionality to convert in the opposite direction too.

Information gained from comments in the questionnaire was also considered when designing the application:

- Possibility to handle a large amount of data during a long period of time. For example being able to refer to a bug that was reported three years ago or being able to report a bug that cannot be solved during the next few years.

- Possibility to invite clients to the project. The clients should be able to see the progress and move cards to completed but their rights should be limited so that they cannot change things everywhere.

- It should be possible to change the task board columns with respect to name and number.

- There should be a simple feature card without a user story.

- Cards should automatically get a human friendly ID number.

- There should be a view that shows the stories and tasks in a list and not only a board with cards.

- Possibility to link and set rules between cards, e.g. a task cannot be started before another task is done. Cards could be linked with is related to, is dependent on, etc.

- Possibility to enable/disable rules. Two respondents want both the possibility to have no rules and the possibility to add a rule that only allows client or Product owner to move the cards to completed.

- Two respondents commented that the rules should enlighten and encourage the users to follow the rules rather than enforcing them. The rules should encourage the users to follow the process that has been decided by the team.

- Rules that limit the columns to which a card can be moved. For example, a card should not be moved directly from "Todo" to "Done".

### 4.2.2 User story workshop

The user story workshop resulted in nine user roles and 33 user stories regarding the product backlog, sprints and support issues (for details, see Appendix A). The stories are not prioritized by MoSCoW, in this case "should" means that the workshop participants think that the story should be included. Identified user roles were:

- Product owner - responsible for prioritizing the product backlog.

- Scrum master - responsible for keeping the backlog updated, create sprint backlog and ensure that everyone is following the decided process.
- Team leader - responsible for a team on the workplace, where the team members can work with multiple projects, either together or separately.

- Team member - developer or designer that works with the project.

- Intern tester - responsible for ensuring that the product owner’s intent is understood by everyone and that the formulation of definition of done is suitable. An intern tester is also responsible for reviewing code and functionality before delivery.

- Extern tester - a client that ensures that delivered functions are what they wanted to have.

- External rapporteur - a client reporting bugs and other problems.

- Internal observer - someone from the company that is not working with development of the product but needs to view the progress of the project, e.g. VD or seller.

- External observer - someone outside of the company that has an interest in viewing the progress, e.g. client or stakeholder.

A user should be able to view the backlog and see detailed information about a backlog item. They should also be able to add, edit and prioritize backlog items. Furthermore, users should be able to set definition of ready on backlog items so that it is possible to know if the items are ready to work on. It should be possible to merge multiple backlogs to get an overview of multiple projects. Users should also be able to sort and filter items so that they can find relevant backlog items. It should be possible to order backlog items hierarchically to see how they are related, and order them in epics/releases with story mapping to get an overview of the user flow.

Regarding sprints, it should be possible for users to create sprints with goal and determined length, and select which stories to solve during the sprint. Users should be able to get an overview of the sprint. It should be possible to start a sprint and use a task board so that the development team can work with the selected stories and visualize the work flow. When working with the cards on the task board, team members should be able to assign themselves to a card. Team leaders should also be able to assign a card to someone else so that they can distribute the work. It should also be possible to end the sprint and move not finished cards to the backlog so that they can be included in another sprint.

Clients should be able to report support issues using a template so that all necessary information is received by the development team. A scrum master should be able to see new support issues in an inbox where support issues can be accepted and moved to the product backlog so that it can be planned and then solved by the development team. There should be a view over support issues so that clients can follow the progress of the support issues they have reported, and the development team can keep track of all the support issues. It should be possible to filter support issues so that users can find relevant issues depending on certain attributes. Team members should be able to see on the card that it is a support issue and if the client has a support agreement so that it can be handled in a certain way.
5 Competitor analysis

Three popular or otherwise relevant project management applications, that are free to use or have a free trial, were evaluated. The purpose of the evaluation was to find good design solutions and which problems they have that should be improved in the prototype.

A common way to evaluate user interfaces is with empirical methods, having users testing the interface [26]. However, using real users in sufficient numbers can be difficult and expensive, a way to avoid this is by the use of inspection, which is a set of methods where evaluators inspect the interface [26]. One inspection method is heuristic evaluation, meaning that usability specialists evaluate the interface based on heuristics, i.e. usability principles [26, 27].

5.1 Method

The applications were studied with focus on heuristic evaluation, based on requirements obtained from the requirement analysis in chapter 4 and guidelines by Molich and Nielsen [28]:

- Simple and natural dialogue - dialogues should contain only relevant information, which should appear in a natural order.
- Speak user’s language - the dialogue should be expressed in a language familiar to the user rather than system-oriented terms.
- Minimize user memory load - the user should not have to remember information from one part to another. Instructions should be simple and easily retrievable.
- Be consistent - a system action should be achievable by a particular user action and users should not have to wonder whether different words, situations, or actions means the same thing.
- Provide feedback - the user should be informed about what is going on by obtaining appropriate feedback within reasonable time.
- Provide clearly marked exits - a user should not be captured in situations with no visible escape.
- Provide shortcuts - include shortcuts that can be used by experienced user.
- Provide good error message - the error messages should blame the problem on the system rather than criticising the user. Furthermore, it should provide the user with exact information about the problem and provide the user with suggestions of what to do next.
- Prevent errors - the system should prevent errors from occurring. An error is here seen as an unsuccessful action, with or without error message.

When evaluating the applications, basic tasks such as creating a new project, and common tasks based on the interviews in [chapter 3] were performed. The functionality and features were compared with the described guidelines and requirements gathered in [chapter 4].

5.2 Result

Trello [10] was evaluated since all development teams at Codemill have used it and four out of five teams were mainly using Trello as project management application during this work. Pivotal Tracker [29] and ScrumDesk [30] were also evaluated since they were found popular when searching for best project management applications using Google. Furthermore they all have some major differences, which provides a wider range of possible solutions.

5.2.1 Trello

Figure 5.2.1: A started project in Trello where the cards represents tasks that are moved between columns, representing the state of the cards. The field to the right provides a menu with project options, and a list of activities in the project.

Trello visualizes the work flow with a task board, where cards are moved between different lists (see Figure 5.2.1). This is the only view over a project, which makes story mapping and sprint planning difficult. There is a menu on the right side with options to add members, change background, filter cards, stickers and Power-Ups (which you can purchase to get additional functionality). This menu field also shows all activity on the board, such as created lists and cards, moved cards and people assigned to cards.
Figure 5.2.2: An open card in Trello provides a view of the content with buttons to the right with options to add content and perform actions.

There is only one type of card without possibility to create sub cards. When creating a new card the user is allowed to set a title, assign members, labels and position of the card. When a card is created it is possible click on it to view and add further information on the card, such as description, check lists, attachments, comments and activities (see Figure 5.2.2). There are two ways to edit a card. When hovering a card an edit icon appears, which allows the user to change title, labels, members and due date, move card and archive card. To make other changes, the user has to close this edit view to open the card by clicking on it.

Figure 5.2.3: The start view in Trello shows the user’s current projects and teams and enables the user to create new projects and teams. The picture shows what it looks like when creating a new project.

One strength with Trello is that it is easy to use. It is easy to switch board, create lists and
tasks, and "drag and drop" makes it easy to move tasks in and between lists. Members can be added to the project but the only rights to choose between are "administrator" or "normal". The difference between them is that an administrator has the rights to change settings for the board. When creating a new project you can name the project and choose to include it in a group, which is a group of people and boards (see Figure 5.2.3). The new board is completely empty except for a field with the text "Add a list".

Trello uses simple and natural dialogues that contains relevant information, starting with the most important and continues deeper into details and less important information. Trello provides multiple language to choose between which enabled the system to speak the user’s language. No need to remember information between different views and dialogues was found, which indicates minimized user memory load.

<table>
<thead>
<tr>
<th><strong>Table 2 Advantages and disadvantages with Trello based on the requirement analysis and guidelines by Molich and Nielsen</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advantages</strong></td>
</tr>
<tr>
<td><strong>Functionality</strong></td>
</tr>
<tr>
<td>There is a task board with flexible columns</td>
</tr>
<tr>
<td>Cards can be filtered by search phrase, labels, assigned to and due date</td>
</tr>
<tr>
<td>Cards can be assigned to members</td>
</tr>
<tr>
<td>Users can see and invite members</td>
</tr>
<tr>
<td>Cards can be moved with drag and drop</td>
</tr>
<tr>
<td>A card can have title, description, label, check list for task, attachments, comments and activities</td>
</tr>
<tr>
<td>A card has functionality to copy, move and remove card, copy link and subscribe to card</td>
</tr>
<tr>
<td><strong>Design heuristics</strong></td>
</tr>
<tr>
<td>Consistent with formulations, icons and looks</td>
</tr>
<tr>
<td>Often provides good feedback</td>
</tr>
<tr>
<td>Simple and natural dialogue</td>
</tr>
<tr>
<td>Provides clearly marked exits</td>
</tr>
<tr>
<td>Provides shortcuts for commonly used actions</td>
</tr>
<tr>
<td>Prevents errors</td>
</tr>
<tr>
<td>There is only one type of cards without possibility to create sub cards</td>
</tr>
</tbody>
</table>

Trello is consistent with formulations, icons and looks, except for saving changes. When clicking on the task board behind a window where changes have been made, the changes
are sometimes automatically saved while they are discarded other times. Trello does not provide any feedback that indicates whether the changes have been saved or not. Clicking on the list name allows the user to change the name directly in the title field and changes are automatically saved without a visible way to undo changes. This differs from similar scenarios, such as creating a list or editing a card, where all changes have to be saved by the user. However, when editing an open card all changes are automatically saved. The esc key can either close a window without saving changes, cancel editing or save editing. If you for example open a card and click on the field to write a comment you have to press esc twice in order to close the window.

Trello provides feedback about what you are about to do and what you have just done. If the user would be unsure of what has happened, there are activity logs for projects and cards that provide information about what has happened. The well recognised cross is always used to close an open dialogue. Trello provides 25 different keyboard shortcuts for commonly used actions, e.g. esc to close window, space to assign card to yourself.

The only error message shown during the test was that Trello is trying to reconnect and that changes may not have been saved. This message did not provide any suggestions of what to do. Trello does not provide many rules which means that there are not many actions that can cause error messages. It is for example possible to set due date to a passed date. Trello prevents mistakes by remembering information entered by the user if a dialogue is closed without being saved. The results from the heuristic evaluation is summarized in Table 2.

Figure 5.2.4: The view over a newly created project in Pivotal Tracker

5.2.2 Pivotal Tracker

When creating a new project, the user has to enter project name, account and project privacy (private/public). When the project is created, a view with a field, containing two empty panels named Current/Backlog and Icebox appears (see Figure 5.2.4).

There is a main menu at the top of the application, containing Stories, Analytics, Settings and Members, and a sub menu at the left side. When opening a project the main view is "Stories", which shows a sub menu with the items "Add story", "My Work", "Current/Backlog", "Icebox", "Done", "Epics", "Labels" and "Project History". When clicking
on an item in the sub menu, the corresponding panel appears or disappears from the panel field.

The stories are shown as expandable list items in the panels, which is the only view over the stories. When clicking "Add story" an expanded story is shown in the panel (see Figure 5.2.5). Here you can see and edit name, type (feature, bug, core, release), points, requester, owners, followers, text fields for description and labels, check list with tasks, activity (comments). When you save the story it is collapsed to minimal view where you can see type, story points, if there are any tasks, owner and state.

When expanding an already created story, it looks like when you add a new item but now the story has an ID, state (see Figure 5.2.6). Furthermore, some functions are "unlocked", you can copy a link or ID of the story, clone story, view history and delete story. You can also switch to full view of the story, which is the same as the create/edit window but larger.
When hovering the left part of a collapsed story, a small window appears with all its tasks, request information and ID (see Figure 5.2.7). On the right side of the collapsed story there are bars to set estimate of the story. However, it does not give any information about what it is and it does not give any feedback about what has happened when it is clicked. When clicked, the button "Start" appears where the bars used to be. Each story has to be estimated before it is started.

There is only one level of cards, which is stories. It is however possible to add tasks in a check list and group stories in epics. The panel Epics allows the user to create epics and add stories to them. The stories can either be created directly from the epic or from another panel by label it with the name of the epic. You can plan to do a release by adding a release story but it is not possible to create releases, containing iterations.

The panel Current/Backlog can be split into separate panels and the panels can be cloned but it is not possible to add, remove and rename panels. When splitting Current/Backlog, Current contains all the stories that should be done in the current iteration whereas Backlog contains all other stories that should be done at some point. The user can choose between automatic or manual iteration planning. The automatic planning uses velocity (number of story points that can be done during a sprint) to move the top of the Backlog, with a story point sum equal to the velocity. When planning manually you can use drag and drop to move the chosen cards to Current. When using automatic planning, it does not seem to be possible to make any changes. Nothing happens with drag and drop but there is no explanation of why it does not work.

When starting to work on the stories, you pick a story from Current and press the "Start" button, which will change to "Finish". It is also possible to start a story from the Icebox or Backlog, this will move the story to the Current Panel. In "My Work" you can see all stories you are working on. Each item in this list has a "Reveal" option which shows the item in the Current panel. When the story is done you press the "Finish" button, which turns into "Deliver". When a story is delivered you can choose to accept or reject it. If the story is accepted it becomes green in the Current/Backlog. You can choose to hide accepted stories from the backlog. If you reject a story, a small window appears where you can leave a
comment. When rejected, the button "Restart" appears with a red circle, similar to a record icon. When hovering or entering the story, any comment about the rejection is showed in the activity of the story. When editing a story it is possible to change state to anything, this way it is possible to correct mistakes. The "Done" panel contains all the completed iterations.

Analytics shows the progress of a project or iteration, such as statistics with graphic presentation of epics, releases, velocity (see Figure 5.2.8). In settings you can change settings for the project, for example enable/disable tasks, decide iteration attributes such as start date, iteration length, point scale and initial velocity. Iteration time can be set to 1, 2, 3 or 4 weeks and it is not possible to change this date. It is possible to invite people as owner, member or viewer in a project.

The information given is relevant and appear in a natural order and the language is natural with familiar terms. Everything is shown in the same view so there is no need to remember information between views and dialogues, which minimizes user memory load.

The system is mostly consistent in looks and functionality between different views and situations. However, when an item is expanded, there is a button with the text "Close" to the story while the cross is used everywhere else. Furthermore, this cross is located on the left side for panels and on the right side for open items.

The system provides poor feedback in many situations. For example, when a story was moved to another position in the backlog it disappeared with no feedback indicating what happened. Furthermore, when trying to add a story in the backlog nothing seemed to happen and no feedback was provided. The story was later found in current iteration, which was closed when the story was added. Another problem is the bars shown on some collapsed stories, which do not explain what they do. Sometimes when clicking on them, a button with the text "Start" appears. Other times when clicking on the bars on a story in current iteration, it was moved to the backlog together with some other stories without any explanation of what happened. When trying to move stories from backlog to work, it did not work and the system did not provide any information about what happened.
In most situations, clearly marked exits are provided with the well known close cross or a button with the text “Close”. However, when entering personal settings there was no clear exit to return to the project. The only way found was to click on the Pivotal tracker logo to choose to enter the project again.

### Table 3

Advantages and disadvantages with Pivotal Tracker based on the requirement analysis and guidelines by Molich and Nielsen

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionality</strong></td>
<td></td>
</tr>
<tr>
<td>There is a backlog in form of a list</td>
<td>It is not possible to change the columns in the task board</td>
</tr>
<tr>
<td>It is possible to group stories in epics</td>
<td>There is only list view over the project progress</td>
</tr>
<tr>
<td>Possibility to expand/collapse items to get more details of a story</td>
<td>It is not possible to filter/sort cards</td>
</tr>
<tr>
<td>Stories have ID, title, description, estimation, label, tasks, attachments, comments and activities, request date</td>
<td>Stories are missing definition of ready, definition of done, acceptance criteria and dependencies</td>
</tr>
<tr>
<td>There is support for sprints</td>
<td>There is little possibility to customize iterations</td>
</tr>
<tr>
<td>Iterations can have defined length and velocity</td>
<td>It is not possible to create releases</td>
</tr>
<tr>
<td>A story has functionality to copy story, remove story, copy link and follow story</td>
<td>It is not possible to define a goal of an iteration</td>
</tr>
<tr>
<td>Possibility to invite members with limited rights</td>
<td>It is difficult to see who accepted the story</td>
</tr>
<tr>
<td>Stories can be moved with drag and drop</td>
<td>Stories are missing the functionality to move the story</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Design heuristics</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Speaks user’s language</td>
<td>Provides poor feedback, e.g. when moving a story it may disappear without any feedback about what happened</td>
</tr>
<tr>
<td>Minimizes user memory load by presenting all necessary information in the same view</td>
<td>Lacking ability to prevent errors since it is not always clear what will happen after an action</td>
</tr>
<tr>
<td>The error messages provides information about what went wrong and what to do next</td>
<td>Missing error messages in some situations where nothing seems to happen</td>
</tr>
<tr>
<td>Mostly consistent in looks and functionality</td>
<td>There are different ways to close objects in different situations</td>
</tr>
<tr>
<td>Often provides clearly marked exits</td>
<td>It is unclear how to return to project from profile</td>
</tr>
<tr>
<td>Provides shortcuts</td>
<td>There are no shortcuts for some often used actions, such as assign to and start</td>
</tr>
<tr>
<td>Relevant information that appears in a natural order</td>
<td></td>
</tr>
</tbody>
</table>
15 shortcuts are provided, of which 9 of them toggle panels. The other shortcuts are used for help, save open story or edited comment, search, add story/epic. There are no shortcuts for other frequently used actions, such as, assign to me or start/finish story.

The error messages that appeared during the tests provided information about what went wrong and what to do next. For example when trying to start an unestimated story, the message "Estimate then start" appears together with a drop down menu where the story can be estimated. In other situations, such as, when trying to move stories between panels without success no error message or feedback was given about the problem.

Some errors were caused because it is not always clear what can be done in different situations or what happens when the user tries to perform a task, therefore the system lacks ability to prevent errors. Furthermore, when closing an edited item, the changes are automatically cancelled without asking the user if they want to save the changes, which can cause unfortunate mistakes. A summary from the heuristic evaluation is seen in Table 3.

5.2.3 ScrumDesk

![ScrumDesk Window](image)

**Figure 5.2.9:** Window for creating a new project in ScrumDesk. The user has to enter project name and organisation and has the option to add description and set project status.

ScrumDesk is, as the name suggests, made specifically to support Scrum and all its events. You have to create a backlog and plan a sprint before the work can start. It can therefore be difficult to use for a team that does not follow Scrum. When you start the application, the first view shows all your projects. When creating a new project, a window appears where the user has to enter project name and organisation (see Figure 5.2.9). It is also possible to add description and toggle between private/public. When creating a project and clicking
Figure 5.2.10: A started story map in ScrumDesk, where stories are ordered in epics to visualize the user flow of a system. The panel to the right shows an open story that provides detailed information about a clicked story.

outside of the window, it is cancelled and the user has to start over. There is a menu on the left side where the user can change view, and an action bar on the top with actions for the current view.

When a new project is created, a view over a story map appears. The board is empty except a card with the text "Unassigned Epic". When clicking on the card nothing happens, at this point it is unclear what to do. When moving the pointer around on the screen a field with the text "ADD NEW BACKLOG ITEM" appears below the unassigned epic card. There is an arrow indicating some options on the right side of this field, however, when hovering it, it is shown that it cannot be used. When creating a new story, a sidebar appears on the right side, where all information about the story can be entered (see Figure 5.2.10). Stories can be of the type user story, technical, bug, research, card or improvement. A story can be assigned to an epic and a theme. Stories can, just like epics, be prioritized after MoSCow, assigned a business value, risk value, kano, tags and attachments. In contrast to an epic, a story can be assigned to a release and a sprint, have acceptance criteria and a list of task to perform.

Changes made in stories and tasks are automatically saved and cannot be cancelled or undone after the edited field has lost focus. There is no visible way to undo changes, when for example editing description, but for more experienced users the common command ctrl + z will undo the last change as long as the focus has remained on that specific field.

Some problems occurred when entering information about the story. A due date is not necessary, if however, a due date is set it can be changed but not removed. Furthermore, release and sprint are clickable but cannot be set, nothing at all happens when clicking on them and no information is given about it. A similar problem occurred when trying to set epic when no epic had been created. A drop down menu appeared with the text "No options" and the clickable text "Manage epics in Story Map", however, nothing happened when clicking on the text.

Epics are created horizontally on the board and the contained stories are organised vertically
under corresponding epic. Epics can be created either with a plus button in the top action bar or with an arrow on an already created epic to add a new epic to the right of that card. However, this arrow is not on the card for unassigned epics. Furthermore, epics can be collapsed and expanded, which is not possible with the unassigned epic.

Figure 5.2.11: The product backlog with an opened backlog item, providing detailed information, in ScrumDesk.

Figure 5.2.12: The plan tab in ScrumDesk provides a tool where a user plans the work. The left part shows an overview of the project’s releases and iterations. The progress of a selected release or sprint is presented in the middle and detailed information is presented to the right.

When entering Backlog, all stories created in the story map are now shown in a list that can be filtered, sorted and grouped by different attributes. Backlog items can be sorted by ID, title, priority, business value, effort, MoSCoW, kano, state, estimated, sprint and release. Furthermore, they can be filtered by team members, iteration, status, type, priority, search word, among others. In this view, backlog items are added with a plus in the action bar.
This will add an empty row at the bottom of the list and the same side bar as in story map appears where information about the story can be entered (see Figure 5.2.11). The empty row is filled with correct information as soon as an attribute has been changed.

When the backlog is created, the user can start to plan the project. When clicking plan in the menu, "Backlog" followed by a box named "Icebox" is shown (see Figure 5.2.12). In the upper left corner there is a plus, which you click on to add a new release. When the release is added it appears in a new field, together with its own icebox, below the backlog field. Below the new release, a clickable text "Add sprint" appears. When the text is clicked, a pop-up appears where a new sprint can be created with goal, start date, due date. The new sprint appears in the release field.

The user can choose to view up to three sprints/iceboxes side by side. Each sprint icebox contains stories that can be dragged between the fields to move them to another sprint. When moving a story with drag and drop, it is not possible to see which story you are holding since it is only represented with a small dark circle. The background of the whole story field, where the story will be dropped, is shadowed. It is possible to mark and move multiple items, however, a sidebar with settings appears, which makes it difficult to move the items with drag and drop. It is however possible to move the marked items by setting release and sprint in the side bar.

![Work tab visualizes the work flow using swimlanes that represent each backlog item and the related tasks.](image)

If you go to "Work" before you have planned the work, it says "There are no sprints defined, add some in Plan!". However, it is not possible to create a sprint if you have not created a release. There is not even any information about how to add sprints before at least one release is created. When desired stories are moved to correct sprint you can use "Work" in the menu, a helping text appears "Please select a sprint to work with". When choosing a sprint with planned stories, a task board with swimlanes appears (see Figure 5.2.13). This board contains all stories planned in the chosen sprint in a column named "Backlog", followed by all its tasks in a "Todo" column. New backlog items can be added with the recognized plus button in the action bar and tasks can be added by clicking on an "ADD TASK" field that appears in a table box on hover.
Figure 5.2.14: Retrospective in ScrumDesk with support for multiple methods. The cards can for example represent improvement proposals, with support for voting and state, such as todo, in progress or rejected.

At this point, it is not possible to move a task to any of the other columns, ”In progress” or ”Done” and there is no explanation of why it does not work. When clicking on an empty space of the board, the sprint information appears in a side bar, with the text ”PLANNING”, followed by an arrow, behind the title. When clicking on PLANNING a drop down menu with the option start appears. The text changed to ”RUNNING” and a burndown chart appeared in the side bar. Now that the sprint is running, it is possible to move the tasks. When dragging a task, a shadow appears where the task will be placed if it is dropped. The tasks can be moved to any row, and thereby to another story. When dragging a task, it is unclear where it was taken from. It is however indicated by an ADD TASK card that remains in its place until the task is dropped.

ScrumDesk also has support for Retrospectives. The retrospective is based on cards with improvement ideas that can be moved freely on a board, similar to index cards on a wall (see Figure 5.2.14). There are some different techniques to choose between which provide different types of cards that can be mixed freely on the board. The members can rate the proposals by clicking on one of the three stars on the card. Each card has state todo, in progress, rejected or done. The state is shown in the bottom right corner of the card and looks like it can be manipulated directly from the card, like the rating. However, clicking on the state opens the card in the side bar.

There is also a view called RCA in the menu, where the user can create a diagram to illustrate cause and effect of problems. When entering this view it is completely empty except for the text ”Diagrams” followed by an arrow. When clicking on it, a small window appears where diagrams can be opened by dragging it to stage. When dragging the diagram to the board a small box with the text ”Problem” appears. When clicking on it, title, description
Table 4 Advantages and disadvantages with ScrumDesk based on the requirement analysis and guidelines by Molich and Nielsen

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advantages</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>There is support for sprints</td>
<td></td>
<td>Very adapted to Scrum, with little support for other methods</td>
</tr>
<tr>
<td>Sprints have goal and due date</td>
<td></td>
<td>Sprints do not have demo date</td>
</tr>
<tr>
<td>Backlog can be created using list and story mapping</td>
<td></td>
<td>There is no support for support issues</td>
</tr>
<tr>
<td>Stories have most of the desired attributes</td>
<td></td>
<td>Stories are missing Definition of done and dependencies</td>
</tr>
<tr>
<td>Tasks have all desired attributes</td>
<td></td>
<td>Bug card does not have information about environment, how to reproduce or contact person</td>
</tr>
<tr>
<td>Backlog items and tasks have most of the desired actions</td>
<td></td>
<td>It is not possible to subscribe to a backlog item or task</td>
</tr>
<tr>
<td>Backlog items can be filtered and sorted by all desired properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backlog items can be prioritized by placement in list</td>
<td></td>
<td>Backlog cannot be prioritized from Work</td>
</tr>
<tr>
<td>Possible to use themes and epics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Items can be moved with drag and drop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>There is a view over project members</td>
<td></td>
<td></td>
</tr>
<tr>
<td>It is possible see who has moved an item</td>
<td></td>
<td></td>
</tr>
<tr>
<td>There are no rules for how to move cards in task board</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Design heuristics</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple and natural dialogue</td>
<td></td>
<td>Does not minimize user memory load since the user sometimes has to switch back and forth between different views to solve a problem</td>
</tr>
<tr>
<td>Speaks user’s language</td>
<td></td>
<td>Does not provide clearly marked exits, e.g. template window does not have an action to close, it can only be closed by clicking on ”Templates” again, which is not obvious</td>
</tr>
<tr>
<td>Consistent with looks and functions in different views and situations</td>
<td></td>
<td>Does not provide shortcuts</td>
</tr>
<tr>
<td>Provides good feedback to let the user know what is about to happen and what has just happened</td>
<td></td>
<td>Error messages are confusing and does not provide any information about how to solve the problem</td>
</tr>
<tr>
<td>There is a task board that visualizes the work flow</td>
<td></td>
<td>The system poorly prevents errors</td>
</tr>
</tbody>
</table>
and category can be edited in the side bar. Furthermore, a small box with the text "Why?" appears in a corner of the problem box. When the why box is clicked a new problem box appears with an arrow to the original problem box.

Relevant information is shown in a natural order, and in a natural language. It is not obvious that you have to follow a specific work flow, which often causes the user to miss a step, and instructions for how to solve the problem are presented in another view. This causes problems with the user memory load that requires going back and forth between different views to solve the problem.

The system is mostly consistent, with similar looks and functionality in different situations. However, drag and drop in Plan looks different than in Story map and Work, where it is clearer where the dragged item will be placed. ScrumDesk provides feedback about what is about to happen and what has just happened.

It is not obvious how to close windows from the action bar. If the user clicks on Templates in the action bar when the filter window is open, it will open a new window on top of the filter window and it does not disappear when clicking on the filter window. The window has no action to close either, it has to be closed by clicking on the Templates action again. The filter window on the other hand, has buttons to cancel and close filter. However, on some screens, it cannot be seen without scrolling horizontally to the end of the window.

No shortcuts were found in ScrumDesk. The error messages shown were confusing and did not provide suggestions of what to do next. It is possible to create a sprint with start date earlier than today but a strange error message appears when trying to start the sprint: "Fail: Due Datetime due date not greater than start date". Furthermore, when choosing a sprint with no planned stories the message says "No backlog items match current filter criterias!" which suggests that the problem is in the selected filter. The results of the evaluation is summarized in Table 4. 
6 Prototype

Prototyping is essential to create successful software and user experience [31]. User-centered design (UCD) was used to develop a prototype with focus on the end users. An important part of UCD is iteration of design solutions and prototype testing on actual users [12].

6.1 Method

It is important to receive user feedback on different concepts early in the development process to avoid committing too early to one design and to avoid high costs of late changes [12, 32, 33]. Therefore, two design concepts were developed and evaluated using medium fidelity prototypes that represents the basic layout and functions, which were tested on potential users. As Rubin and Chisnell [12] suggest, a refined prototype was created with the best attributes of both prototypes.

6.1.1 Design concepts

Medium fidelity prototypes of two different concepts were created in Axure RP [34]. The purpose of the prototypes was to explore and test different approaches on potential users to evaluate which approach works best and avoid committing too early to one design. With this purpose in mind, a prototype should have enough functionality to address the test objective, any other functionality is not necessary [12].

The prototypes showed the different views of the application and the intended functionality. The prototypes had enough functionality to show how the system would work and what you can do with it. Grayscale was used to group objects in order to provide a better overview and still have the unpolished low fidelity look, which can make the users more open to providing constructive feedback [33].

The functionality and features included in the prototypes were based on the requirement analysis presented in chapter [4]. All the requirements with priority must and should were included in both design concepts. Some requirements with priority could were also included in the designs if it did not interfere with higher priority requirements or need too much resources. Requirements with priority Won’t was only implemented if it facilitated higher prioritized requirements.

6.1.2 Usability tests

Usability testing is an important part of UCD, that should be performed early in the design process by testing prototypes with actual users [12]. Dumas and Salzman describe characteristics of valid usability tests, which were included when designing the test. These
characteristics state that the primary goal is to improve the usability of a product, typical tasks for the product are performed by end users or potential end users of the product who are usually asked to think aloud as they perform the tasks. Furthermore they state that the tests are observed and recorded, the data is analysed and results in recommended changes to fix diagnosed problems in the product [35].

The tasks and questions in the usability test were based on the most important functions and features identified in the requirement analysis in chapter 4. A semi-structured approach was used to collect qualitative data. When performing user tests during the design phase, the goal is to understand why the users act as they do and ask for their ideas about how to improve confusing areas [12]. An exploratory comparison test was also conducted on the concepts in order to uncover strengths and weaknesses in the concepts and how the system can be improved. Exploratory comparison tests aim to answer which style is the most effective, which the best and worst features of the approaches are, which the main stumbling blocks for the user are, and for which tasks the user may need more help [12].

It has been showed that five participants will uncover in average 80 % of the usability problems found by 20 participants and these five participants uncover most of the severe problems [33]. After 5-8 test sessions, the sessions start to become unproductive since the same problems are repeated without finding many new problems [33]. Therefore six participants were chosen for this usability test. The participants were potential users who are employees at Codemill. They were specifically chosen from different teams using different agile methods and with different team and project roles to receive a wider range of feedback. Each concept was tested by three participants and the differences between the concepts were compared by all six participants. Before the usability tests, a pilot test was conducted on one potential user to ensure that participants can understand the questions and receive necessary feedback when performing a task.

The test was conducted with one participant at a time. The participant was placed in front of a laptop with a mouse, whereafter they were introduced to the subject, the purpose of the test and the procedure. Lookback [36] was used to record the activity on the screen, speech and face expressions with the participant’s consent. Each participant tested one of the concepts on the laptop, after which they compared both concepts side by side to explain which elements they like better and why. The participant was asked to think aloud as they performed the tasks since it is vital to explore the users thought process in this stage [12].

As suggested by Rubin and Chisnell [12], the participant was given real life tasks to perform or show how they would like to perform it. For example order the backlog after a specific attribute, edit a backlog item or start working on a task. Furthermore, the participant was asked questions about how they perceived the views and their opinions about it. One type of question used was what the participant expected would happen when they clicked on a certain element, whereafter they clicked and was asked if they how it met their expectations. From the start view, for example, the participant was asked ”What do you expect will happen if you click on Project A?”'. The purpose of this type of question was to evaluate if it is clear what the functions do or in this case to decide which view to show when you enter a project. In other cases the participant was given a short scenario leading to another view. One such scenario was to create a sprint. The participant was asked how they would do to create and plan a sprint. They were told that they could navigate to any view they like, which should lead the participant to the Plan view. When all the tasks had been performed, the participant was asked what they thought was the major difficulties with the concept and how they they
thought it could be made easier.

The session ended with side-by-side comparison of the two concepts to understand why aspects of one concept are more favourable than the other concept [12]. Printed pictures of the major differences between the concepts were placed in front of the participant (see Appendix B). The participant compared one view at a time, with both concepts side by side and was asked to explain why they think one approach is better than the other. The test ended with letting the participant view the prototype freely and comment or ask questions about the design.

### 6.1.3 Refinements

In order to increase the usability of the system, the design was refined based on the usability tests. As suggested by Rubin and Chisnell [12], the most favoured parts from the concepts were merged into a refined design, and problem areas diagnosed during the usability tests were changed to better meet the users expectations. The prototype was made more realistic to give a better picture of how it would look and work.

A workshop was held together with two UX-designers/developers and one developer/team leader/Scrum master. The purpose was to identify and discuss solutions to problems diagnosed in the usability test which did not have a clear solution. The goal of the workshop was to discover solutions to:

1. do (sprint) planning in Product backlog.
2. add themes, epics and backlog items in Story map. It should be possible to add the items at any position in a row (themes and epics) or column (backlog items).

The workshop started with introducing the participants to the work with the prototype, the purpose of the workshop and the procedure. The participants were then given a printed screenshot of Product backlog and were introduced to the problem regarding (sprint) planning in Product backlog. All participants sketched their ideas on the printed screenshot for 10 minutes, whereafter they presented and discussed their solutions for 5-10 minutes. This procedure was then repeated for the second problem, with a printed screenshot of Story map.

### 6.2 Result

#### 6.2.1 Design concepts

Since the concepts have many similarities, the first concept is described thoroughly in Concept 1 and the differences are presented in Concept 2.
Figure 6.2.1: The start view of the application shows projects the user is involved in, and enables the user to create new projects.

Figure 6.2.2: All backlog items shown in list view with detailed information about each item.

Concept 1

The start view presents all the projects the user is involved in and enables the user to create a new project (see Figure 6.2.1). The icons in the upper, right corner show which user is logged in, and if the user has any notifications, such as being assigned to a task or any activities of an issue that the user has subscribed to. The profile can be edited by clicking on the avatar, and personal settings, such as language, time, etcetera can be changed when clicking on the settings icon. These icons are fixed through the whole system.
Figure 6.2.3: Clicking on the filter icon reveals filter options for different attributes.

Figure 6.2.4: When clicking on a backlog item, information about that item appears in a sidebar.
(a) Details tab shows detailed information about the backlog item.

(b) Comments tab shows comments on the backlog item and enables the user to add comments.

(c) Activities tab shows all activities regarding the backlog item.

(d) Editing mode shows all attributes except for ID as editable fields.

Figure 6.2.5: The different views of an open backlog item.
When clicking on a project, the product backlog appears (see Figure 6.2.2). The product backlog displays all backlog items with ID, state, title, assignees, theme, epic, release, sprint, estimate, MoSCoW and creation date. Items with tasks can be expanded by clicking on the plus, which will reveal the tasks. The backlog items can be sorted by clicking on the corresponding attribute in the list above. Furthermore, they can be filtered by clicking on the filtering icon, which will reveal a dialogue with filtering options (see Figure 6.2.3).

When clicking on a backlog item, it will be opened in a side bar to the right (see Figure 6.2.4). Important properties such as ID, title, state and tags are always shown on the top of the card together with actions to move, copy, copy link, edit, delete and close card. The bottom bar of the card shows assignees to the left and a clickable icon to the right for assigning the card to a member.

This concept uses tabs to switch between detailed information, comments and activities (see Figure 6.2.5). Details shows description, tasks, acceptance criteria (see Figure 6.2.5a). Comments shows all comments written on the card, who wrote them and when (see Figure 6.2.5b). It is possible to mention users in comments to send a notification to that user. Activities shows all changes and activities on the card, who was responsible for it and when it occurred (see Figure 6.2.5c). To edit information on card, the user clicks on the pen in the upper right corner, which will change the appearance of the card so that all information fields editable (see Figure 6.2.5d) and buttons to cancel or save changes appear. To add a new backlog item, the user clicks on the plus button, which reveals a sidebar with an empty card in editable mode.

![Figure 6.2.6: Story map enables the user to order backlog items hierarchically by theme and epic horizontally, and by iteration vertically](image)

The story map is a graphic tool for planning the work and creating themes, epics and stories in a hierarchical order (see Figure 6.2.6). When creating items here, they are automatically added to the backlog and vice versa. The top row presents themes followed by its contained epics on the second row. Both themes and epics are ordered horizontally on the board while the stories are organized vertically below respective epic, ordered by release. When hovering a column, a card with a plus appears at the bottom of the column, indicating that a new card can be created there by clicking on the plus card. A similar card appears at the end of the theme row and epic row when hovered.
Figure 6.2.7: This view enables the user to plan iterations and choose which backlog items to do during the iteration.

Figure 6.2.8: Swimlanes is the default view for the task board. Each row represents one backlog item and its tasks.

Plan in the menu is a tool for planning releases and sprints (see Figure 6.2.7). This view consists of three panels. The left panel presents information about the current sprint and enables the user to create new releases and sprints. Clicking on a release or sprint will open it in the middle panel. The middle panel displays information about a release or sprint and a list of all backlog items planned in this release/sprint. The sprint backlog can be sorted and filtered using the action buttons above the list and the items can be prioritized with drag and drop. The panel to the right displays the product backlog, which can be manipulated in the same way as the sprint backlog. To plan which backlog items to perform during a sprint,
the backlog items can be moved from the product backlog to the sprint backlog with drag and drop.

Task board, which is the tool used by a team member when working on tasks, visualizes the work flow. There are four different views to visualize the tasks in Task board, which are swimlanes, columns, lists and single list. The default view is swimlanes, which shows the relation between a backlog item and its tasks (see Figure 6.2.8). One row in the table represents one story, which is represented by a larger card at the top of the Todo column. Its tasks are placed below it until a user moves it to another column. The tasks can be moved between the columns with drag and drop. When all the tasks in a row have been moved to Done (or a column with state done), the Backlog item will also be moved to Done. When hovering a task, a clickable icon appears in the bottom left corner to assign the task to somebody. When hovering a segment in the table, a plus card appears where a task can be created and automatically added to the backlog item in that row.

To be able to adapt the task board to the work process of a team and project, columns can be added, removed, renamed and the status on a column can be changed between not started, ready to work on, in progress, pending and done. However, there has to be one column with state not started and one with state done so that the swimlanes work properly and a finished backlog item is moved to the column with state done.

![Figure 6.2.9: Columns view presents the tasks as individual objects that is moved between the columns](image-url)
Figure 6.2.10: Lists view presents tasks as list items with one row for ID and title

The columns view only shows the tasks, as separate objects, organized in columns that represent their current state (see Figure 6.2.9). Columns and tasks have the same functionality in this view as in swimlanes. The third view (see Figure 6.2.10) visualizes the tasks as list objects in each column in order to provide an overview of a large amount of tasks. The fourth view is single list, which looks like the product backlog.

Figure 6.2.11: View over support issues, with a side bar to report new issues

Support presents reported support issues and enables a client to report new support issues (see Figure 6.2.11). The support issues are presented in a list, similar to Backlog, however, the attributes shown in the lists are not the same. The support list shows information about ID, state, title, support agreement, rapporteur, assignees, report date and deadline. A side
bar shows a form to report a new issue. The form contains information necessary for a bug, based on the requirement analysis in [chapter 4]. The client can enter a subject, choose project, describe the problem in free text, add relevant environment information, attachments and contact information to a person that can be contacted if any more information is needed.

**Figure 6.2.12:** The product backlog in the second concept. An open backlog item is presented as a pop-op

**Figure 6.2.13:** Swimlanes in this concept presents the backlog items in their own column before their tasks are presented in the corresponding row

**Concept 2**

One of the main differences between the concepts is that the second concept uses pop-ups (see Figure 6.2.12 instead of side bars. Furthermore, when opening a story, the pop-up presents all information in the same view instead of using tabs for different kinds of information. Another different seen in Figure 6.2.12 is that this concept does not use the
Figure 6.2.14: The diagram presents the number of participants that performed the tasks in the backlog without any errors or confusion.

explaining sort list above the backlog items. Instead action buttons are used to sort the backlog items.

Another difference between the concepts is how the swimlanes in Task board looks and works. In this concept, the backlog item is located in their own column followed by columns for their tasks (see Figure 6.2.13). In this concept, the state of a backlog item is indicated by the coloured circle and a finished backlog item will remain in the same column with a green circle.

6.2.2 Usability tests

Since some parts are the same in both concepts, they were tested by all six participants while the differences were tested by three participants and then compared by all six participants. The result is therefore divided into two groups where there are differences between the concepts, and presented as one group where there is no difference between the concepts.

All six participants found it easy to understand the start view. When clicking on a project, the participants expected to get a new view with project details or issues to solve, for example visualised on an agile board. When entering a project, all participants found it easy to switch to another project.

Backlog

How well the participants performed the tasks in the Backlog is illustrated by Figure 6.2.14. All participants from both groups could easily find and understand the plus button to add a new backlog item. All participants testing concept 1 understood how to sort the objects after a specific attribute, however, one participant was worried that state would sort the
objects after alphabetical order on status in stead of work flow order. The participants
testing concept 2 were looking for a list over the items, which was only used in concept 1.
When realising that this approach was not possible, they saw the sort buttons and clicked
on the one with the tooltip sort. One participant mentioned that they would like four states
for sorting assignees, that is after number/me combined with ascending/descending.

To filter the items, one participant testing concept 1 at first wanted to click on the header
above that attribute but realised that it would sort the items. The participant looked around
and found the filter button. The other five participants quickly recognized the filter button
and clicked on it. All participants could easily use the filter, however, one participant would
like to have all the filters shown above the backlog instead of clicking a button to show it.

All the participants from both groups wanted to be able to prioritise the items using drag
and drop, three of them however (two testing concept 1 and one testing concept 2), were not
sure if it would work since there were no number for priority or that it would feel strange if
the list was sorted after something else than priority. One participant proposed that making
the items separate objects could clarify that they can be dragged.

To edit an item from the backlog, all six participants would open the item by clicking on
that item, which after they would click on the field to change. Three participants (one using
concept 1 and two using concept 2) thought that is was unclear whether it would work or
not. After clicking on the field, the participants looked at the action buttons in the right
corner and found the edit pen.

**Story map**

When clicking on Story map in the menu, the participants expected to see an agile board,
backlog items in another format than a list, something with user stories, multiple abstrac-
tion levels of what is needed in the product, or stories ordered in epics and releases. Two
participants had clear expectations of how the story map would look and work. They both
agreed that stories should be ordered hierarchically and placed in releases and the stories
should only show a title. Furthermore, one of them expects it to be very graphic, like a story
map on a wall and possibility to choose colors.
The chart shows how the users expected to be able to add a new release. The sum in the chart exceeds the number of participants because some participants answered with two alternatives.

One participant was not used to this organisation of stories and had some problems to understand the flow of the stories. All of the participants found it easy to add a new card, one participant, however, was a little confused by the smaller cards at the end of a field. Another participant said that he might want to add a new card by pressing ‘n’ but started to wonder where that card would be added. One participant stated that in a large project, it can be difficult to find the add theme card. Two participants wondered if it was possible to add support issues in the story map. One of these participant would like to have a theme for support issues to ease release planning. To add a new release, all six participants searched around the left part of a line. They expected to create a new release by clicking on a plus near the line, use a plus button in the action bar, clicking on an existing release/backlog or drag a story on a line (see Figure 6.2.15).
Plan

When the participants were asked where they would navigate and how they would do to create a new sprint, they expected it to be done in either Plan, Backlog or Story map (see Figure 6.2.16). One participant said that they would like to do more planning in the story map. This participant wanted to create a new sprint by marking desired stories and click on a plus that would show somewhere, or they would add a new release or sprint in the story map and drop the stories there. Without possibility to create sprints in the story map, this participant would try to create sprints in the backlog or task board. This participant said that plan looked like a calendar which was not of interest and was therefore ignored. Two participants wanted to create new sprints from the backlog but realised that it was not possible and found plan in the menu. The other three participants immediately clicked on plan. In plan, the participants felt that they were on the right place, however, they needed some time to find the option to add sprint. One participant looked for a plus button like the button to add backlog item.

Task board

Some participants were curious about the view buttons and navigated through the views as soon as the entered the task board. One participant said that it was not that clear what they meant. When asked to change the view, the participants clicked at these buttons immediately. One of the participants hovered the icons and used the tooltip before clicking on the different views. One participant was a little confused by the list view since it looks like the backlog. Furthermore, one participant wanted to see assignees in the list view and thought that it was unnecessary with description in the columns view.

The participants liked the possibility to change view in the task board. When working with
development they would prefer swimlanes or columns but two participants said that lists can be good when planning. Swimlanes was the most popular choice as default view (favoured by one participant testing concept 1, and two participants testing concept 2), followed by columns (two participants testing concept 1) and personal setting (one participant testing concept 2). One participant also mentioned that the start view of Agile board could be a dashboard to show how the project is going, for example say that a card has been in the same state for a while and asked if you have forgotten to move it.

To start working on a task, all participants would drag the card to “In progress” and either be automatically assigned or, be able to assign yourself quickly, for example, by pressing space or a single button click. Two participants want to be automatically assigned while two others think it can be problematic since the team leader sometimes moves a card for someone else and assign them to distribute the work. One participant says that the card should not have to be dropped in the correct row, it should stay in its current row but move to the column it was dropped in. All participants easily found and understood the assign button.

To add a new task, one participant was going to click on the column menu but noticed the big plus card and clicked there instead. Four participants immediately moved the cursor to where they wanted to add the card and clicked on the appearing plus card. One participant discovered how to add a new card before the interviewer could ask them to do this task.

Support

Three participants thought that the support in the menu was to get help using the system, it might be easier to understand if it was called "Issues". One participant mentioned that they might have wanted to add a support card in the task board before they clicked on Support in the menu. When asked how they would do to report a support issue all the other participants were fast to click on support.

All participants found it easy to fill in the report form. None of the participants found it necessary to add a description for attachments, however, it could help referring to the attachments in the description if they were named. Two participants wanted to divide the description into free text and a list of how to reproduce the problem step by step. One of the participants would like something similar to the environment solution, where you add one step and an empty step appears below.

The information in the report form should be well formulated and explained so anyone knows what information is needed and why. The environment information should be optional since it is not always relevant. One participant wants to be able to adapt the retrieved information to a specific project. Furthermore, it can be good to know which user found the problem and priority/impact that represents how urgent the issue is to the client.

One participant thinks it would be good if the support view is adapted to the the person watching it so that you see relevant information, for example depending on if you are a client, Scrum master or team member. This participant also mentioned that they want to be able to search for old issues to see if the same problem or something similar has been reported earlier. Another participant want to be able to comment on and reply a support issue. Furthermore, this participant wants to see if the issue is added to a release, and they want a connection between similar issues, for example if it can be solved with an issue from
Difficulties

Two participants testing concept 1 thought that the largest difficulty was to understand the story map. Both participants said that they are not used to this method of creating user stories. The third participant testing concept 1 thought that the most difficult task was to start a sprint, and it was unclear if the items in the backlog could be moved. This participant wanted a more graphic way to plan sprints.

Two of the participants testing concept 2 thought it was hard to sort the backlog items. They also thought that it was too many steps to edit a story, and that it was not clear whether it was possible to edit the story directly when you have opened it. The third participant testing concept 2 thought it was confusing with many similar views and that it was difficult to understand Plan.

Comparison

The most preferred approach from the comparison is shown in Figure 6.2.17. All of the participants from both groups thought that the swimlanes view was easier to understand in concept 2 since it was clearer that one row represent one story. One participant however, would like to have concept 2 but with the story as a header on the top of the row. One participant wants the story to fill the whole vertical space in its column while another participant wanted the story to be free, outside of the table.

All the participants liked the backlog in concept 1 better since it is clearer and easier to sort. This was especially clear for the participants testing concept 2. The participants found
it difficult to decide whether side bar or pop-up was best since they both can be good in different situations. However, five of the participants favoured one over the other. The three participants testing concept 1, favoured the pop-up in concept 2. Of the participants testing concept 2, one favoured pop-up, one favoured side bar, and one could not decide which one is best. Two of the participants testing concept 1 said that they would like the possibility to toggle between side bar and pop-up. The advantages with the pop-up is that you have focus on what you are doing, it feels like you can close the window at any time by pressing escape button. Furthermore, pop-up works well with large screens and half screen, with another window open on the side. The advantages with the side bar is that it feels easy to switch between different stories and tasks, and you can see everything in the background.

Two participants testing concept 1 favoured all information in one view over the tabs because it is easier to scroll through the information than clicking on tabs which requires more precision. One participant testing concept 2 favoured the tabs. The remaining three participants could not decide which was better.

6.2.3 Refinements

A refined prototype was created based on the user tests and the sketch workshop held to find different solutions to solve problems discovered during the usability test. The refined prototype contains the best parts from both concepts, and has a more realistic look than earlier prototypes.

Sketch workshop

The workshop resulted in different solutions to do sprint planning in the backlog, and to add themes, epics and backlog items in Story map. The printed pictures of the Product backlog and Story map with idea sketches that were gathered after the workshop can be seen in Appendix C. When the ideas were presented and discussed, the following solutions were identified to plan releases and sprints in Product backlog:

- Options above the list of backlog items to create and handle releases and sprints.
- Tabs that enables switching between view over epics/stories/releases/sprints, where the existing view would be the story view. Clicking on sprint tab would reveal a list of existing sprints with related attributes instead of the backlog items.
- Plus icons in the attributes field to add release, sprint, theme, epic by clicking on the plus in the corresponding column.
- Group backlog items after release/sprint so that new releases and sprints can be added at the desired position. Backlog items can then be moved between sprints.
- Direct manipulation of attributes in the list so that backlog items can easily be moved to a sprint. This could be used for all attributes to enable faster editing of backlog items. At the end of the list there could be a row with editable fields where a new backlog item can be created by inserting attributes of the new backlog item.
- A side bar that can be pulled out, where the user can handle sprints and drag items from the backlog list to a release/sprint to add it to that iteration.
The following solutions were identified to create new cards in Story map:

- Plus, dots or thin card between existing cards to indicate that a new card can be created there. This indicator could either be visible at all times at all places where a new card can be created, or be revealed when hovering that area.
- Every card has options to insert a card before or after itself.
- An empty card at the end of each row/column that can be clicked to create a card at the end or dragged to create a card at the position where it is dropped.
- Cards similar to desktop icons where a card can be created at any position, and multiple cards can be selected by creating a square around the cards to select. The user can set size of a release/sprint space by dragging the line up or down.

A suggestion delivered by a participant was to clarify which epics and stories belong to a theme. For example by making a theme card as wide as the width of its contained epics and stories, or by separating the segments for themes.

**Refined design**

The refined prototype has more realistic backlog items and tasks, synchronized through the whole system. Members have real profile pictures instead of an icon of a person, and the text and values were changed to look more like a real project. The menu was clarified with feedback that indicates which view the user has entered. Furthermore some menu items were given new titles or icons, e.g. Backlog was changed to Product backlog, Task board changed to Agile board and Support changed to Support issues with a new icon.

Since plan was not clear to all test participants and 50% of them wanted to plan releases and sprints in Backlog or Story map, Plan was removed and this functionality was added in Backlog and Story map. This approach requires fewer steps and gathers more information and functionality in the same place.

Since the user tests did not reveal any problems with the start view, the only changes made was to make the design more realistic, with images and image placeholders (see Figure 6.2.18). In the refined prototype, all views provides the status of backlog items and tasks in both color and text to support colour blindness.
Figure 6.2.18: When starting the application, the user is presented with a view over the projects they are involved. It is also possible to create a new project from here.

Figure 6.2.19: The product backlog presents all backlog items in a list that can be prioritized, sorted and filtered.

Based on the usability tests, the refined design of the backlog (see Figure 6.2.19) includes the header above the backlog items since all participants favoured this approach. Furthermore all the participants testing concept 2, without the header, was searching for a header when sorting the items. Since some participants thought it was unclear whether the items could be moved, they were separated to clarify that they are individual objects that can be moved.
Figure 6.2.20: An open backlog item is presented as a pop-up with all information gathered in the same view.

Figure 6.2.21: An open task presented as a pop-up, similar to the backlog item.
Figure 6.2.22: Swimlanes is the default view in Agile board. Each row represents one backlog item.

The pop-ups without tabs were used to show stories and other items since a larger group of participant favoured this approach (see Figure 6.2.20). From the usability tests presented in section 6.2.2, it was clear that the users want to be able to edit the items faster. Therefore, the item can be edited directly in the list and when the item is opened it can be edited by clicking on the attribute to edit, which will change into, for example, a text field that indicates that the field can be edited. Furthermore, when a card has been edited, the text ”All changes have been saved” appears with an undo button so that changes made since the card was opened can be undone. Enabling editing directly requires more actions in the same view. Therefore, the actions was split so that actions regarding the card, such as copy link or close, were placed in the upper right corner. Actions regarding the content, such as add check box list or add attachments, were placed in the lower right corner. An open task looks similar to an open backlog item with a few differences to provide information that is relevant for a task (see Figure 6.2.21). The task has a link to the backlog item containing the task so that the parent card can easily be reached.

Agile board shows a header of the current sprint, which can be clicked to open information, similar to backlog items and tasks. Furthermore there are options behind it, where the user can select, start or stop a sprint. In all views for the Agile board, the columns show number of tasks within to better support Kanban. Product backlog and Support issues can be pulled out in Agile board from the menu so that backlog items can be easily moved into the current iteration without having to change view.

Based on the comparison test, the refined design included the swimlanes from concept 2, where the story is shown in a separate column since all the test participants thought it was easier to understand (see Figure 6.2.22). However, some changes were made to take better advantage of the space. Backlog items cover the whole height of their row. Furthermore, they show more information the higher they are.

Tasks covers the whole column width instead of presenting two tasks side by side. This approach provides a better overview when many columns are used. However, when all tasks of a backlog item is placed in the same column it may require a large space vertically.
Therefore, backlog items can be collapsed and expanded. A collapsed backlog item only shows state, ID, title, tags and assignees, all tasks are hidden. A task card presents the same information as a collapsed backlog item card. The height of a task card is adapted to its content so that a card with a long title is larger than a card with a small title. However, the card shows maximum three lines of the task title, any longer title will end the third lines with three dots to indicate that it is more to read.

Figure 6.2.23: Column view presents the tasks as independent objects in columns representing their state.

Column view presents all the tasks in columns as independent objects (see Figure 6.2.23). The task cards look the same as in the swimlanes view. List view is similar to columns,
however, a task only presents the title on a single row with assignees below to provide a better overview of a large amount of tasks (see Figure 6.2.24).

The refined design of Story map is presented in Figure 6.2.25. The theme card covers the whole column width to clarify that everything in that column belongs to the theme. Cards can be added anywhere in a row/column and not only at the end. When hovering a section in the table, pluses appears where a new card can be added in that segment.

Since all the participants in the usability tests searched for functionality to add new releases around the left side of the lines, it was placed there. When the the left part of the line is hovered, a plus followed by ”Add sprint” appears under the line. Another line appears underneath to indicate where the sprint will be placed.

Figure 6.2.26: Support presents a view over support issues reported by clients.
Support issues is similar to Product backlog but the information shown in the list differs (see Figure 6.2.26). A support issue shows information about ID, state, title, rapporteur, assignees, impact, support agreement, planned and report date. Impact was added in the refinement to enable the clients to report how urgent the problem is. An open support issue is similar to backlog items and tasks but the information differs. Besides the changes in the list, each support issue can also have information about how to reproduce the problem, environment (operating system, browser, etcetera) and contact information to a person that can be contacted for more information about the problem (see Figure 6.2.27).

Some changes were also made when reporting an issue (see Figure 6.2.28). How to reproduce the problem was extracted from the description. How to reproduce the problem is instead described in a bullet list starting with one bullet and visible option to add new row. The option user was added to the environment since it can be important to know for which user the problem occurred. Furthermore a short text was added to the contact field, explaining that it is for contacting someone in case any more information is needed.
Figure 6.2.28: A form presented as a pop-up where the clients can report support issues, such as bugs and other problems
7 Discussion

The competitor analysis revealed strengths in all tested applications. However, none of them meets the requirements compiled in the requirement analysis, chapter 4. Trello is easy and fast to use but lacks important functionality, such as support for product backlog, sprints, multiple level and types of cards. Users are forced to come up with their own solutions which may not be optimal but still preferred over a more complex and difficult system. Pivotal tracker on the other hand has support for backlog, sprints and epics but it provides a process that has to be followed strictly which can force a development team to adapt their process after the restrictions of the application. It would be better if the application could be customized after the team’s development process. ScrumDesk provides many of the required functions with good support for software development using Scrum, such as sprints, backlog, story mapping, task board and retrospectives. However, the system is too adapted to Scrum to be suitable for a workplace where the development teams use different methods. An attempt was made to fill the gap by designing an application with the required functionality that is easy and fast to use and does not force the users to follow a specific method.

The result from the analysis of the development teams in chapter 3 indicates that one of the most important features of a project management application is efficiency. In this study, interview respondents preferred Trello over other applications, even though it is missing important functionality, because it is easy and fast to use. A system like this has to be easy and fast to use, both for experienced and inexperienced users so that developers want to use it every day and inexperienced external clients can use it to report support issues and prioritize backlog items. Shortcuts can help experienced users perform tasks even quicker.

Different development teams in a workplace might have varying processes even though they follow the same agile method and use the same project management application. In one team, a member unassign from a task to show that it needs to be reviewed while other teams use multiple columns for the review process. From this study, it is difficult to find a reason to the varying development processes. It could be that the teams have found different solutions to solve problems caused by using a too simple tool that does not support functionality required by agile methods such as Scrum and Kanban. It could also be that they simply prefer this method.

A development team should not be forced to adapt their process to limitations of a tool, which means that the application has to be flexible. However, adding too many possibilities to the application may cause a more complex and difficult application. When designing the prototype, an attempt to achieve this balance was made by initially implementing the most important features that all the teams desires. Then additional functionality desired by only some of the teams was implemented if it did not compromise the efficiency of the system. Another way to achieve flexibility without causing the system to become complex by adding too much functionality could be by avoiding unnecessary rules of usage. For example provide the user with the possibility to start working on created tasks without
creating a sprint or iteration first, and possibility to start working on a not estimated task. Rules like these could however help the team follow a process that has been established by the team. A solution could be to enable the user to add such rules to a project in the project settings.

The survey reveals that most of the users do not want rules and rights to control what they can do with the system. However, two respondents commented that they would like the possibility to enable/disable rules. Furthermore, a team leader and scrum master wanted rules for what different members can do, which gives more control to the team leader/Scrum master. The view to handle members and their right was not implemented in this prototype due to the limited time. The problem with members rights could be solved by using roles such as team leader, scrum master, team member, product owner and viewer. Since many respondents in the survey did not want rules to control what the user is allowed to do, it might be a good idea to give team members full rights as standard and let a team leader limit the rights for different roles or specific members.

The result of this work was a prototype that fills the gap between the examined applications, where the users’ needs are in focus. It is difficult to say whether this prototype is everything they need now since the refined prototype has not been tested. However, after more user tests and refinements I believe that this application will meet their needs. The findings in this work are based on Codemill’s needs, however, they could also be considered by others with similar needs.

### 7.1 Limitations

All parts and functionality of the system were not designed, due to time constraints. The work was focused on the most important parts based on requirement analysis, and functionality needed to support Scrum and Kanban since those are the methods that should be followed by the development teams at Codemill.

Because of the time limitation, the refined prototype has not been tested. The concepts were tested on employees at Codemill, which covered most of the end users. However the refined prototype should be tested on potential clients with less experience of similar applications.

### 7.2 Future work

There is still a lot of work to do, both with the prototype and further investigation. Handling of projects, members and settings has not been included in this prototype, and requires further investigation to decide which functionality should be included before starting to design and implement these views.

One respondent mentioned in an interview that they liked Trello’s mobile client. Such a mobile client could be a great possibility for future work. However, it would require a thorough investigation to gain knowledge about what the users would want to do from a mobile device.

Since one of the most important properties is that it is fast and easy to use, it may be good to use shortcuts for the more experienced users. Participants in the interviews and user test
expressed that it should go fast to assign oneself to a task, possibly by using a shortcut. There are also other actions that could be performed faster with shortcuts. However, further investigation is required to decide which actions should have shortcuts. A good start could be to analyse which actions are mostly used and which actions could be performed significantly faster with shortcuts.
8 Conclusions

The development process in a workplace can vary a lot between different development teams and projects. All teams at Codemill follow a method inspired by Scrum and/or Kanban, where close collaboration with the client is important to achieve frequent delivery of the highest prioritised tasks. Another similarity between the teams is that they all have a list of backlog items, from where backlog items are broken down into tasks that are presented on a task board, visualizing the work flow. Even though the teams’ development processes are inspired by the same agile method and have many similarities, the execution may be very different. For example, all teams use some kind of iterations but the type of iteration and how they use them vary between the teams. One team frequently moves backlog items from the backlog to the task board during an iteration while other teams have a sprint meeting where they plan which backlog items should be done during the coming iteration and move those backlog items to the task board. It is important that the system is flexible enough to handle these differences so that the teams can use it the way they want to instead of having to adapt their process after the system.

The results in this work indicate that the users want a system that is fast and easy to use both for experienced and inexperienced users. The system should also be flexible in the sense that the users should be able to adapt the task board, and the usage should not be restricted by rules and roles. However, some users want to have selectable rules so that a scrum master or team leader can ensure that a certain process is followed.

The project management applications examined in the concurrent analysis all have their strengths, however, none of them meets the needs of the development teams at Codemill. The most popular was Trello because it is fast and easy to use but it is missing a lot of important functionality. ScrumDesk works best when following Scrum, and requires that every step is followed strictly. Pivotal Tracker is more complex, which can make it more difficult to use, but it is still missing important functionality such as a graphic task board. It is important to find a balance between simplicity and enabling the user to do what and how they want in the system.

The aim of this master thesis was to develop a prototype of a project management application adapted for Codemill. The result of this work was a prototype that seeks to fill the gap between the examined applications, where the users’ needs are in focus. However, the prototype needs further testing and refinement in order to ensure that the users needs are met.
References


[34] “Axure.” [Online]. Available: [https://www.axure.com](https://www.axure.com/)

A Requirements

A.1 Requirements

This document contains the requirements for the project management application. The requirements were compiled from a user story workshop with one workshop leader and three end users, and a questionnaire answered by 17 end users.

A.1.1 User stories

In the user story workshop, the following roles were identified and used when creating the user stories:

- Product owner - responsible for prioritizing the product backlog.
- Scrum master - responsible for keeping the backlog updated, create sprint backlog and ensure that everyone is following the decided process.
- Team leader - responsible for a team on the workplace, where the team members can work with multiple projects, either together or separately.
- Team member - developer or designer that works with the project.
- Intern tester - responsible for ensuring that the product owner’s intent is understood by everyone and that the formulation of definition of done is suitable. an intern tester is also responsible for reviewing code and functionality before delivery.
- Extern tester - a client that ensures that delivered functions are what they wanted to have.
- External rapporteur - a client reporting bugs and other problems.
- Internal observer - someone from the company that is not working with development of the product but needs to view the progress of the project, e.g. VD or seller.
- External observer - someone outside of the company that has an interest in viewing the progress, e.g. client or stakeholder.

Backlog:

- As PO/SM/TM I want to add items to the backlog so that I can add problems to solve
- As PO/SM/TM I want to edit items so that I can adapt to changes
- As PO/SM/TL I want to view the backlog so that I can get an overview of all items to solve
• As a user I want to expand and collapse items so that I get a good overview and can see details of a specific item
• As a user I want to filter the items so that I can see only relevant items
• As TL I want to merge multiple backlogs to get an overview of multiple projects
• As PO/SM I want to create epics so that I can categorize stories
• As PO I want to sort items so that I can see the items after a specific attribute
• As PO I want to order items hierarchically so that I can see how the stories are related
• As PO/SM I want to order cards in epics/releases with story mapping so that I get an overview of the user flow
• As PO/SM/TM I want to prioritize cards so that the most important cards are solved first
• As PO/SM I want to create releases so that I can plan the work
• As PO/SM I want “Definition of ready” for cards so that I know if a card is ready to add in a sprint

Sprints and iterations:

• As SM I want to create a Sprint with a goal so that the team can work with stories to achieve that goal
• As PO I want to decide length of the sprints so that frequent delivery is ensured
• As PO I want to choose which stories to solve during the sprint so that the team can deliver desired stories
• As SM I want to start a sprint so that the team can start working on the stories
• As SM/TM I want an agile board so that I can visualize the work flow
• As TM I want to move cards between different states so that it is possible to follow the progress of the card
• As SM/TL I want to prioritize the sprint items so that the most important things are solved first
• As TM I want to assign myself to a card so that everyone can see that I am responsible for that card
• As SM I want to assign a card to someone else so that I can distribute the work
• As PO/SM/TM I want to see an overview of the sprint so that I can see the progress of the sprint
• As SM I want to end a sprint so that implemented work will be delivered on time
• As a PO/SM I want to move the cards not solved during the sprint back to the backlog so that we can add them to another sprint
Support issues:

- As PO/ER I want to create a support card so that I can report bugs and other problems
- As SM I want a template for support cards so that all necessary information is given from an external rapporteur
- As SM I want to get support cards in an inbox so that I can see new support cards
- As PO/SM/TM I want to accept a support card so that it is moved to the backlog if it is approved
- As SM/TM/ER I want a view over the support cards so that I can keep track of them
- As a user I want to filter the cards so that I only see the relevant cards
- As SM/TM I want to see if a card is a support card so that I know that I have to handle it in a certain way
- As SM I want to see if there is a support agreement with the client so that I know how to handle the issue
- As PO/ER I want to follow a support card to know what is happening to it

A.1.2 Questionnaire result

The following tables (Table 1-10) presents the questionnaire results. Responses were valued as -2 (not at all important), -1 (somewhat important), 0 (no opinion/do not know), 1 (important), and 2 (very important). The MoSCoW priority was determined by the average value and the percentage of positive responses (important or very important).

**Must:** average value of at least 1, and/or at least 75 % positive responses

**Should:** average value of at least 0, and/or at least 50 % positive responses

**Could:** average value of at least -1, and/or at least 25 % positive responses

**Won’t:** average value below -1, and less than 25 % positive responses
Table 5 The table presents the respondents’ desire for general functions and features. Avg is average value (between -2 and 2), positive responses is percentage of responses for important/very important, and MoSCoW represents the priority of the feature (Must/Should/Could/Won’t).

<table>
<thead>
<tr>
<th>Function</th>
<th>Avg</th>
<th>Positive responses</th>
<th>MoSCoW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Backlog from list</td>
<td>1.24</td>
<td>82.4 %</td>
<td>Must</td>
</tr>
<tr>
<td>Support for Sprints</td>
<td>1.06</td>
<td>76.5 %</td>
<td>Must</td>
</tr>
<tr>
<td>Move cards with drag and drop</td>
<td>1.47</td>
<td>94.1 %</td>
<td>Must</td>
</tr>
<tr>
<td>See members in the project</td>
<td>0.41</td>
<td>64.7 %</td>
<td>Should</td>
</tr>
<tr>
<td>See that a client has approved the implementation of a task</td>
<td>0.29</td>
<td>52.9 %</td>
<td>Should</td>
</tr>
<tr>
<td>Create backlog with Story mapping</td>
<td>-0.35</td>
<td>29.4 %</td>
<td>Could</td>
</tr>
<tr>
<td>Possibility to use intervals for time estimations</td>
<td>0.12</td>
<td>47.1 %</td>
<td>Could</td>
</tr>
<tr>
<td>Possibility to change size and amount of information on the cards on the agile board</td>
<td>0.12</td>
<td>41.2 %</td>
<td>Could</td>
</tr>
<tr>
<td>See activities on the board</td>
<td>0.47</td>
<td>41.2 %</td>
<td>Could</td>
</tr>
<tr>
<td>Auto assign card to a person that moves it to the column in progress</td>
<td>-0.19</td>
<td>41.2 %</td>
<td>Could</td>
</tr>
<tr>
<td>Possibility to create cards from check lists</td>
<td>0.12</td>
<td>47.1 %</td>
<td>Could</td>
</tr>
<tr>
<td>Rules for how the cards can be moved</td>
<td>-0.71</td>
<td>29.4 %</td>
<td>Could</td>
</tr>
<tr>
<td>Integration with time tracking system</td>
<td>-0.82</td>
<td>47.1 %</td>
<td>Could</td>
</tr>
<tr>
<td>Possibility to limit number of cards in a column</td>
<td>-0.94</td>
<td>11.8 %</td>
<td>Won’t</td>
</tr>
<tr>
<td>Provide predefined columns</td>
<td>-1.24</td>
<td>5.9 %</td>
<td>Won’t</td>
</tr>
<tr>
<td>Support for Retrospectives</td>
<td>-1.12</td>
<td>11.8 %</td>
<td>Won’t</td>
</tr>
<tr>
<td>See Burndown chart</td>
<td>-0.9</td>
<td>15.0 %</td>
<td>Won’t</td>
</tr>
<tr>
<td>Roles that controls a members rights</td>
<td>-0.82</td>
<td>23.5 %</td>
<td>Won’t</td>
</tr>
</tbody>
</table>

Table 6 The table presents the respondents’ desire for functions and features regarding sprints. Avg is average value (between -2 and 2), positive responses is percentage of responses for important/very important, and MoSCoW represents the priority of the feature (Must/Should/Could/Won’t).

<table>
<thead>
<tr>
<th>Function</th>
<th>Avg</th>
<th>Positive responses</th>
<th>MoSCoW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint goal</td>
<td>0.41</td>
<td>52.9 %</td>
<td>Should</td>
</tr>
<tr>
<td>Demo date</td>
<td>0.29</td>
<td>52.9 %</td>
<td>Should</td>
</tr>
<tr>
<td>Sprint stop date</td>
<td>0.47</td>
<td>58.8 %</td>
<td>Should</td>
</tr>
<tr>
<td>Possibility to set maximum number of story points for a Sprint</td>
<td>-0.41</td>
<td>29.4 %</td>
<td>Could</td>
</tr>
</tbody>
</table>
### Table 7
The table presents how the respondents want to prioritise backlog items. Avg is average value (between -2 and 2), positive responses is percentage of responses for important/very important, and MoSCoW represents the priority of the feature (Must/Should/Could/Won’t).

<table>
<thead>
<tr>
<th></th>
<th>Positive responses</th>
<th>MoSCoW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Placement in list</td>
<td>88.2%</td>
<td>Must</td>
</tr>
<tr>
<td>Must/should/could/won’t</td>
<td>35.3%</td>
<td>Could</td>
</tr>
<tr>
<td>Number</td>
<td>17.6%</td>
<td>Won’t</td>
</tr>
</tbody>
</table>

### Table 8
The table presents the respondents’ desire to filtrate items after different attributes. Avg is average value (between -2 and 2), positive responses is percentage of responses for important/very important, and MoSCoW represents the priority of the feature (Must/Should/Could/Won’t).

<table>
<thead>
<tr>
<th></th>
<th>Positive responses</th>
<th>MoSCoW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search word</td>
<td>88.2%</td>
<td>Must</td>
</tr>
<tr>
<td>Assigned to</td>
<td>88.2%</td>
<td>Must</td>
</tr>
<tr>
<td>Type/category</td>
<td>76.5%</td>
<td>Must</td>
</tr>
<tr>
<td>Status</td>
<td>88.2%</td>
<td>Must</td>
</tr>
<tr>
<td>Sprint/release</td>
<td>76.5%</td>
<td>Must</td>
</tr>
<tr>
<td>Priority</td>
<td>70.6%</td>
<td>Should</td>
</tr>
<tr>
<td>Requester</td>
<td>47.1%</td>
<td>Could</td>
</tr>
<tr>
<td>Creation date</td>
<td>47.1%</td>
<td>Could</td>
</tr>
<tr>
<td>Estimation</td>
<td>23.5%</td>
<td>Won’t</td>
</tr>
<tr>
<td>Deadline</td>
<td>23.5%</td>
<td>Won’t</td>
</tr>
</tbody>
</table>

### Table 9
The table presents the respondents’ desire to sort items after different attributes. Avg is average value (between -2 and 2), positive responses is percentage of responses for important/very important, and MoSCoW represents the priority of the feature (Must/Should/Could/Won’t).

<table>
<thead>
<tr>
<th></th>
<th>Positive responses</th>
<th>MoSCoW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>94.1%</td>
<td>Must</td>
</tr>
<tr>
<td>Status</td>
<td>76.5%</td>
<td>Must</td>
</tr>
<tr>
<td>Type/category</td>
<td>52.9%</td>
<td>Should</td>
</tr>
<tr>
<td>Sprint/release</td>
<td>70.5%</td>
<td>Should</td>
</tr>
<tr>
<td>Estimation</td>
<td>35.3%</td>
<td>Could</td>
</tr>
<tr>
<td>Creation date</td>
<td>35.3%</td>
<td>Could</td>
</tr>
<tr>
<td>Deadline</td>
<td>41.2%</td>
<td>Could</td>
</tr>
</tbody>
</table>
Table 10 The table presents the respondents’ desires regarding rules for how to move cards in the agile board. Avg is average value (between -2 and 2), positive responses is percentage of responses for important/very important, and MoSCoW represents the priority of the feature (Must/Should/Could/Won’t).

<table>
<thead>
<tr>
<th>Rule Description</th>
<th>Positive responses</th>
<th>MoSCoW</th>
</tr>
</thead>
<tbody>
<tr>
<td>No rules</td>
<td>64.7 %</td>
<td>Should</td>
</tr>
<tr>
<td>A member that implemented a task can not move it from ”Peer review”</td>
<td>0.0 %</td>
<td>Won’t</td>
</tr>
<tr>
<td>One step at a time</td>
<td>5.9 %</td>
<td>Won’t</td>
</tr>
<tr>
<td>Only client/product owner can move card to ”Complete”</td>
<td>17.6 %</td>
<td>Won’t</td>
</tr>
</tbody>
</table>

Table 11 The table presents the respondents’ desire of different attributes for stories. Avg is average value (between -2 and 2), positive responses is percentage of responses for important/very important, and MoSCoW represents the priority of the feature (Must/Should/Could/Won’t).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Avg</th>
<th>Positive responses</th>
<th>MoSCoW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>1.65</td>
<td>100 %</td>
<td>Must</td>
</tr>
<tr>
<td>Free description</td>
<td>1.71</td>
<td>100 %</td>
<td>Must</td>
</tr>
<tr>
<td>Attachments</td>
<td>1.00</td>
<td>70.6 %</td>
<td>Must</td>
</tr>
<tr>
<td>Comments</td>
<td>1.53</td>
<td>94.1 %</td>
<td>Must</td>
</tr>
<tr>
<td>Id</td>
<td>0.53</td>
<td>58.8 %</td>
<td>Should</td>
</tr>
<tr>
<td>Estimation</td>
<td>0.53</td>
<td>64.7 %</td>
<td>Should</td>
</tr>
<tr>
<td>Created (date)</td>
<td>0.18</td>
<td>52.9 %</td>
<td>Should</td>
</tr>
<tr>
<td>Type/category</td>
<td>0.29</td>
<td>58.8 %</td>
<td>Should</td>
</tr>
<tr>
<td>Definition of done</td>
<td>0.00</td>
<td>52.9 %</td>
<td>Should</td>
</tr>
<tr>
<td>Acceptance criteria</td>
<td>0.24</td>
<td>52.9 %</td>
<td>Should</td>
</tr>
<tr>
<td>Dependencies</td>
<td>0.29</td>
<td>52.9 %</td>
<td>Should</td>
</tr>
<tr>
<td>Tasks</td>
<td>0.71</td>
<td>70.6 %</td>
<td>Should</td>
</tr>
<tr>
<td>Activities</td>
<td>0.71</td>
<td>58.8 %</td>
<td>Should</td>
</tr>
<tr>
<td>Requestor</td>
<td>-0.18</td>
<td>41.2 %</td>
<td>Could</td>
</tr>
<tr>
<td>Template for goal/description (As ... I want ... so that ...)</td>
<td>-1.12</td>
<td>5.9 %</td>
<td>Won’t</td>
</tr>
<tr>
<td>How to demo</td>
<td>-0.71</td>
<td>23.5 %</td>
<td>Won’t</td>
</tr>
<tr>
<td>Progress visualisation</td>
<td>-0.29</td>
<td>23.5 %</td>
<td>Won’t</td>
</tr>
</tbody>
</table>
Table 12 The table presents the respondents’ desire of different attributes for task cards. Avg is average value (between -2 and 2), positive responses is percentage of responses for important/very important, and MoSCoW represents the priority of the feature (Must/Should/Could/Won’t).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Avg</th>
<th>Positive responses</th>
<th>MoSCoW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titel</td>
<td>1.18</td>
<td>82.4 %</td>
<td>Must</td>
</tr>
<tr>
<td>Description</td>
<td>1.18</td>
<td>82.4 %</td>
<td>Must</td>
</tr>
<tr>
<td>Referens to story</td>
<td>1.12</td>
<td>76.5 %</td>
<td>Must</td>
</tr>
<tr>
<td>Assigned to</td>
<td>1.65</td>
<td>94.1 %</td>
<td>Must</td>
</tr>
<tr>
<td>Comments</td>
<td>1.47</td>
<td>88.2 %</td>
<td>Must</td>
</tr>
<tr>
<td>Id</td>
<td>0.65</td>
<td>64.7 %</td>
<td>Should</td>
</tr>
<tr>
<td>Attachments</td>
<td>0.76</td>
<td>70.6 %</td>
<td>Should</td>
</tr>
<tr>
<td>Activities</td>
<td>0.71</td>
<td>52.9 %</td>
<td>Should</td>
</tr>
<tr>
<td>Estimate</td>
<td>0.00</td>
<td>47.1 %</td>
<td>Could</td>
</tr>
<tr>
<td>Created (date)</td>
<td>0.18</td>
<td>47.1 %</td>
<td>Could</td>
</tr>
<tr>
<td>Deadline</td>
<td>-0.29</td>
<td>29.4 %</td>
<td>Could</td>
</tr>
<tr>
<td>Type/category</td>
<td>-0.24</td>
<td>29.4 %</td>
<td>Could</td>
</tr>
<tr>
<td>Link to git-branch</td>
<td>-0.12</td>
<td>41.2 %</td>
<td>Could</td>
</tr>
<tr>
<td>Log time</td>
<td>-0.82</td>
<td>17.6 %</td>
<td>Won’t</td>
</tr>
</tbody>
</table>

Table 13 The table presents the respondents’ desire of additional attributes for bug cards. Avg is average value (between -2 and 2), positive responses is percentage of responses for important/very important, and MoSCoW represents the priority of the feature (Must/Should/Could/Won’t).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Avg</th>
<th>Positive responses</th>
<th>MoSCoW</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to reproduce</td>
<td>1.65</td>
<td>94.1 %</td>
<td>Must</td>
</tr>
<tr>
<td>Contact</td>
<td>1.29</td>
<td>88.2 %</td>
<td>Must</td>
</tr>
<tr>
<td>Environment (OS, browser, server, etc.)</td>
<td>0.71</td>
<td>52.9 %</td>
<td>Should</td>
</tr>
</tbody>
</table>
Table 14 The table presents the respondents’ desire of different actions for cards. Avg is average value (between -2 and 2), positive responses is percentage of responses for important/very important, and MoSCoW represents the priority of the feature (Must/Should/Could/Won’t).

<table>
<thead>
<tr>
<th>Avg</th>
<th>Positive responses</th>
<th>MoSCoW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move card</td>
<td>1.65</td>
<td>94.1 %</td>
</tr>
<tr>
<td>Remove card</td>
<td>1.41</td>
<td>88.2 %</td>
</tr>
<tr>
<td>Copy link to card</td>
<td>1.47</td>
<td>88.2 %</td>
</tr>
<tr>
<td>Subscribe to card</td>
<td>1.00</td>
<td>76.5 %</td>
</tr>
<tr>
<td>Convert from task to story</td>
<td>0.24</td>
<td>52.9 %</td>
</tr>
<tr>
<td>Copy card</td>
<td>-0.06</td>
<td>41.2 %</td>
</tr>
<tr>
<td>Convert from story to task</td>
<td>0.00</td>
<td>41.2 %</td>
</tr>
</tbody>
</table>

Comments

The following list functions and features were desired by one or more respondent who requested it in a comments field in the questionnaire. In this case, should is not the MoSCoW priority "Should". The list is not prioritised, all requests should be considered in the design but not necessarily implemented.

- It should be possible to handle a large amount of data during a long period of time. For example being able to refer to a bug that was reported three years ago or being able to report a bug that cannot be solved during the next few years.

- It should be possible to invite clients to the project. The clients should be able to see the progress and move cards to completed but their rights should be limited so that they cannot change things everywhere.

- It should be possible to change the agile board columns with respect to name and number.

- There should be a simple feature card without a story.

- Cards should automatically get a human friendly id number.

- There should be a view that shows the stories and tasks in a list and not only a board with cards.

- It should be possible to link and set rules between cards, e.g. a task cannot be started before another task is done. Cards could be linked with is related to, is dependent on, etc.

- It should be possible to enable/disable rules. Two respondents want both the possibility two have no rules and the possibility to add a rule that only allows client or Product owner to move the cards to completed.

- The rules should enlighten and encourage the users to follow the rules rather than enforcing them. The rules should encourage the users to follow the process that has been decided by the team.
• There should be rules that limits the columns to which a card can be moved. For example, a card should not be moved directly from "Todo" to "Done".
B  Comparison test
B.1 Product backlog, concept 1
B.2 Product backlog, concept 2
B.3 Agile board, concept 1
B.4 Agile board, concept 2
C Sketch workshop

C.1 Adding items in story map
C.2 Sprint planning in Product backlog