CHALMERS
UNIVERSITY OF TECHNOLOGY

Designing an Automated Guided Vehicle configuration software

Master's thesis in Industrial design engineering

PETRA AHNELL & SOFIA ROSENGREN

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE
Division of Design & Human Factors

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

MASTER'S THESIS 2021

Designing an Automated Guided Vehicle configuration software

PETRA AHNELL & SOFIA ROSENGREN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Materials Science
Division of Design & Human Factors
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

Designing an Automated Guided Vehicle configuration software

Examiner: Pontus Wallgren, Department of Industrial and Materials Science

Master of Science Thesis 2021
Department of Industrial and Materials Science
Division of Design and Human Factors
Chalmers University of Technology
SE-412 96 Gothenburg

Cover illustration: Girl sitting on an AGV and using a laptop
Cover made by Sofia Rosengren

Printed by Repro Service Chalmers

Gothenburg, Sweden 2021

Abstract

Automation in terms of automated guided vehicles (AGVs) may sound very modern, but has been around longer than one may think. The first automated guided vehicle systems were invented in the 1950s, and have since become a huge factor for successful logistics in larger warehouses [15]. One of the early adopters of the then-new technology were Kollmorgen Automation (KMA), called NDC at the time. KMA installed the first global application of AGV systems in 1976 at Tetra Pak's factories and have since played a key role in advancing the technology [3].

When implementing an AGV system in a warehouse, or developing a new AGV vehicle, various software are used, and KMA develop some of this software. A crucial step of developing a new vehicle is the vehicle configuration, and it is around this software that this project has revolved. This master's thesis project investigates what factors are important when designing a vehicle configuration tool for AGVs, and presents a design concept for such software. The project has been conducted together with KMA.

In this study, two user groups have been studied: Expert users and beginner users. The expert users of vehicle configuration tools are engineers at KMA and their partner companies. An interview study was conducted to understand their needs and to construct a user journey. In order to get a representation of users less accommodated with the current KMA, beginner users were also investigated. This since one of the major issues with the current software is that they are difficult to learn. In order to understand the usability issues and flaws of the current software tools that beginners might encounter, a usability test was conducted. Based on the user studies, design guidelines were created. A design proposal was also created in the form of a clickable prototype. The prototype was redesigned and refined through several iterations evaluated with usability tests and expert interviews. The result of the project is a set of guidelines for designing these types of software tools, and a design proposal for the KMA case, utilizing said guidelines.

Keywords: AGV, Automated guided vehicle, vehicle configuration, Application engineer, Automation, User interface, User experience, Interaction design, design guidelines, Usability, Usability testing

Acknowledgements

We would like to thank the people involved in this project and that made this work possible.

To our supervisor and examiner Pontus Wallgren at Chalmers University of Technology, for guiding us, for your constructive feedback, and for giving us confidence in the academic parts of this project.

To our supervisor at Kollmorgen, Cecilia Boström, for your tireless support and encouragement, and for sharing your deep UX knowledge. Thank you for the many hours that you have spent on guiding us and setting our project straight, and for your valuable feedback.

To Daniel Lindqvist, for your patience explaining the software systems and technical details, and for your many good ideas.

To all the lovely people at Kollmorgen Automation who welcomed us with open arms, encouraged us throughout the project, and for making us feel part of the KMA team.

To all of our participants for taking the time to participate and for your reflective thoughts and ideas. This project would not have been possible without you.

Thank you.

Glossary

AGV - Automated guided vehicle
CTA - Call to action, such as a button in an interface
Figma - Software tool for constructing clickable interface prototypes
GEW - Geneva emotion wheel
KMA - Kollmorgen Automation
LD - Layout designer, a Kollmorgen path drawing software
Microsoft Teams - Communications software with video call functionality
Miro - Software tool for collaborative work
PLC - Programmable logic controller
Site - Where the AGVs drive, e.g. a warehouse
UI - User interface
UX - User experience
VAD - Vehicle application designer, a Kollmorgen vehicle software
Zoom - Communications software with video call functionality

Contents

1. Introduction

Automation is in many ways a part of our everyday life, even though we may not be aware of it at first glance. For example, when ordering a nice new jacket via an e-commerce website, there is a good chance that the logistics are automated and that your jacket was picked up by an automated guided vehicle (AGV), a load carrier that drives around the warehouse autonomously. Our ways of living, and expectations of companies, shape the development and demands of automation. To meet the future challenges of automated solutions, it is important that the tools used for development are not only functional and technically advanced but that they also are easy to use.

Kollmorgen Automation AB (KMA), a Gothenburg-based company that specializes in providing partner companies with AGV systems and software tools, is aiming for just that: Developing tools for the implementation of autonomous systems. KMA provide tools for drawing paths where the vehicles drive, software that controls the AGV system once in place, and also software for configuring vehicle parameters when creating a new vehicle, among else. The software is used by a vast range of AGV manufacturers, and in many ways, KMA act at the very core of the industry and have made many technological advancements over the years. However, when developing these tools, there has not been much focus on usability or user experience (UX), and much of the functionality is spread out across multiple software. With these two incentives, KMA aim to redesign and rethink their current software.

In this project the result of multiple user studies and usability testing will be presented along with a new software concept used for vehicle configuration, designed according to better accommodate usability principles. The focus has lied on understanding the process of the vehicle configuration software and how users use the current software to create new vehicles, and how that process can be better supported by changes in the software design.

1.1. Design brief

Many of the KMA software have an outdated look to the UI since many of the software were created during the 80's and 90's. Since then, features have been added and functionality reworked. However, the UI design is largely the same as it was when the programs were first developed. The UX and UI have not been considered much throughout the history of the company, and professionals working solely with UX and UI of the KMA software have just recently been hired. Therefore, the UI of these software is in need of a redesign and usability is now a high priority for the company when redesigning the software.

In several of these tools, a vehicle configuration must be done. This is a set-up that must be made for each new vehicle created. Since the different software that KMA provides in many cases do not communicate, the same vehicle configurations are performed in the different software. This causes a user to have to remember to update a vehicle over several programs if a change is made and to type in the same parameters multiple times, which is both time-consuming and can lead to errors. Hence there is a need for a new software where all vehicle configurations are collected in one place, and that has good usability and is designed with the end users' ways of working in mind.

1.2. Users: Application engineers

The primary users of the solution designed in this project are so-called application engineers. Application engineers work as a bridge between the customers, sales team, and in-house engineers. Generally, the users of the KMA software all have good technical knowledge and skills, and a majority have different engineering backgrounds or other technical backgrounds. The users vary in their experience of working with the specific software, and there are users who have worked with the software since they were developed, and there are users who are new at their positions and can in many ways be considered beginner users. Hence, the software has to be developed with both these types of users in mind. Furthermore, the way of working differs between users working at different companies, something that will be addressed later in this text. In the case of the in-house KMA application engineers, they often help partners functioning as customer support or educating partners about the software. They are very well acquainted with the usage of KMA products and also generally know how different partner's application engineers prefer to work when setting up new AGV systems.

1.3. Objectives and aim

The objectives of this project are to understand the needs of the users of the current vehicle configuration software and construct a user journey, to based on this create guidelines on what to consider when creating a vehicle configuration software, and to create a clickable prototype. The aim is to create a prototype that fulfills the user needs and has better usability than the current software.

1.4. Research questions

- What does the user journey look like for application engineers today?
- What are the needs and requirements of the users of today?
- How can the vehicle configuration software be improved in terms of usability?
- What is important to consider when designing a vehicle configuration software?

1.5. Scope

In this project, a graphic software interface concept represented by a clickable prototype has been designed and built using Figma. The software includes the same functionality as the functions tested in the KMA software but does not include all parameters or functions since it is superfluous for the representation. The focus lies on creating a design with better usability for the overall tool, and that is better equipped to match how users prefer to work.

When analyzing the effects of better usability and efficiency, the focus has been on the benefits for the individual user. However, the impacts that the new design could have on the company in terms of sales or savings have not been further analyzed.

The focus has been creating the design of the new tool, and no implementation has been done in code. Since the project focuses on UX and design, the final outcome is a detailed model of the final software concept and not an actual program.

Furthermore, since KMA already has a set colour palette planned to be used in new software, creating a colour palette for the software has not been a part of the project.

Lastly, the project has focused on looking at centralizing the vehicle configuration over three KMA software: Layout Designer (LD), Vehicle Application Designer (VAD), and Reflector Surveyor (RS), with the main focus on the two first mentioned. Although vehicle configuration exists over some other software, the three mentioned were deemed most important and or representative enough by representatives from Kollmorgen.

2. Background

2.1. Automated guided vehicles

An automated guided vehicle (AGV) is a load carrier that can move around autonomously. They are used for material handling in a wide array of applications, ranging from transporting food trays and linen in hospitals to heavy loads in industrial settings. One of the most common applications is transporting pallets in warehouses, using forklift AGVs.

Most commonly, the AGVs drive along predetermined paths that have been designed to fit the environment in which they drive and the tasks that they are to perform. The path drawing is conducted within the KMA software Layout Designer. When the system is set up at a site, the AGV is connected to a master system that provides each vehicle in the fleet with orders on where to drive and what to pick up.

To navigate, the AGVs most commonly use reflectors that have been mounted on the walls at known locations. The vehicle sends out signals via a scanner mounted on top of it and gets a response from the reflectors that can tell it where in the room it is. This technology is known as reflector navigation. However, there is a whole range of navigation methods, such as following a magnetic track in the floor, which requires antennas under the vehicle, as well as "natural" navigation, in which the vehicle scans the environment of the site itself. One vehicle can also use a combination of several methods, for example, reflectors in the open areas of a warehouse, but magnetic track navigation in aisles where it is difficult to mount reflectors.

There is also a safety system on every AGV, making sure it does not collide with a person or an object. The safety system consists of one or multiple sensors and safety fields deriving from each sensor. The sensor uses different safety fields for different situations or "safety cases". Safety cases are defined by, for example, the speed that the vehicle is driving and the steering angle of the vehicle. Typically, each sensor for each case has two fields, a slow down-field and a stop field where the slow down field stretches further from the vehicle. If there is an obstacle very close to the AGV (in its stop-field), it engages the emergency brake and stops immediately. If there is an obstacle in the larger slow down-field, the AGV simply slows down.

AGVs can have different wheel configurations and drives. Steer Drive (SD) is the most common configuration, where the vehicle has one SD front wheel and two fixed back wheels. The SD wheel is both driving the vehicle forward and steering it by controlling the wheel angle. Other common configurations are Quad Drive, with two SD wheels in front of each other, and Differential Drive, with two parallel drive wheels.

2.2. Kollmorgen Automation

Kollmorgen Automation AB (KMA) is a Gothenburg based division of the global technology company Kollmorgen. On a larger scale, Kollmorgen develops technical solutions within a vast field, including aerospace, food, and packaging industry, whereas KMA solely focuses on automation and AGV systems.

The division has a long history of developing AGV electronics and software. It started as a Swedish company named Netzler & Dahlgren Co AB (NDC), which was founded in 1962. In the beginning, they developed specialized electronic equipment for various industries, and

in 1972 they took part in their first AGV project. Since then, the company has grown and now partners with many AGV manufacturers around the world. In 2001 NDC was acquired by a company called Danaher, and the name was changed to Danaher Motion and later Kollmorgen Automation. The name NDC lives on in the form of the product name NDC Solutions.

KMA develops AGV control software, navigation technology, and drive units. They partner with AGV manufacturers to create entire AGV systems. The partner company sell the system to the end customer, usually to be set up in a warehouse or manufacturing facility. Both application engineers from KMA and the partner company can be involved in installing and maintaining the system. Usually, long-time partners set up the system themselves. KMA helps in the cases where the customer is inexperienced, or the system is complicated. They also help with more difficult support cases.

2.3. KMA software

The KMA software includes software to be run by a master computer on site for traffic and flow control, navigation software used by the individual AGV, and software tools for designing and servicing the system. The tools are used by application engineers at KMA and at partner companies.

The tools consist of a large number of software. There are tools for designing the paths the AGVs should follow, programming the behaviour of the vehicles, setting up the reflectors used for navigation, troubleshooting a malfunctioning AGV, simulating the system, and more. Many of the tools need a digital model of the vehicle, and in those cases, the user needs to make a configuration of the vehicle. This means that the user needs to set some parameters relevant for that tool, for example, physical parameters like width or length of the vehicle. Some of the software that include a vehicle configuration are Vehicle Application Designer (VAD), Layout Designer (LD), and Reflector Surveyor (RS).

Vehicle Application Designer
The configuration of the vehicle is done mainly in a software called Vehicle Application Designer (VAD). In this software, parameters are set for drives, scanners, and other objects on the vehicle. There is also a built-in tool for programming the PLC (the control unit) of the vehicle, which controls the vehicle behaviour. From the VAD, a file is downloaded to the vehicle in order to transfer the input from the software to the vehicle.

The software is, in general, based on a two-column tree structure, but there are some differences depending on what view you are in. For example, the PLC programming view looks completely different from the rest of the views since it is based on external software. There are also some views in the software from which you can send commands to the vehicle in real-time while connected to it. These, too, have a different look with not only fields but buttons as well. The scope of this study includes the parameters that can be set from VAD, and to a minimal extent, the PLC programming. It does not include the functionality of sending commands to a connected vehicle since this does not fall under the category of vehicle configuration.

An overview of the software is displayed in Figure 1. On the left-hand side, there are several symbol buttons where you can navigate to different views. The next column contains a tree structure with varying content depending on the selected view. In the main window, a list

of parameters is shown depending on what is selected in the tree structure. At the bottom, there is an explanation shown for the selected parameter.

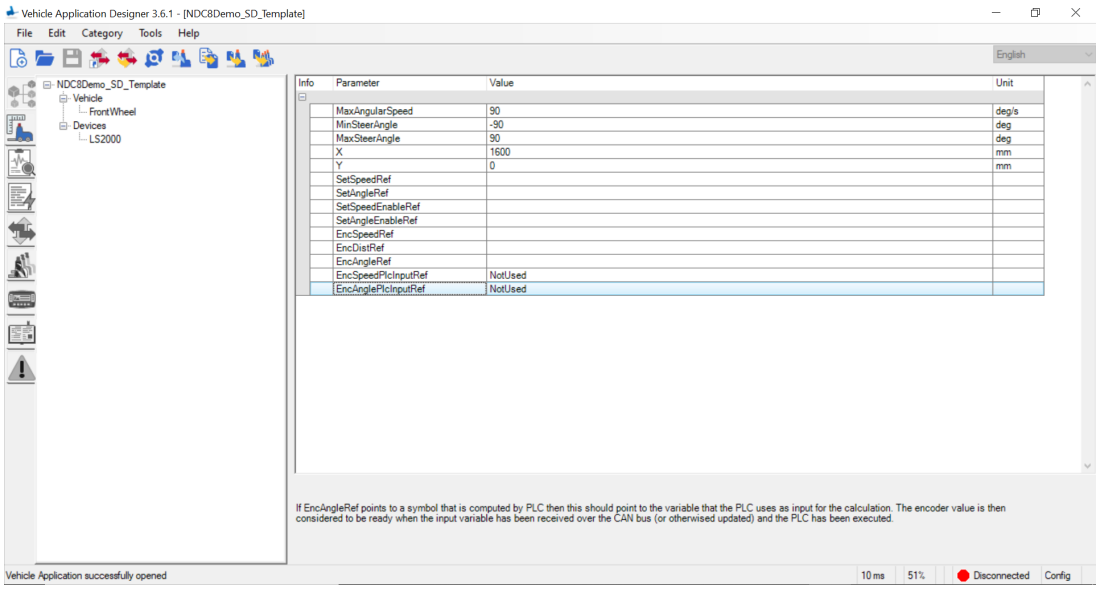

Figure 1: Vehicle Application Designer software.

Layout Designer

In the Layout Designer (LD) software, the routes that the vehicles drive at the site are drawn. In order to do this, a setup of the vehicles is necessary. The type of vehicle in terms of wheel configuration is set, as is the size, wheel placement, safety sensor system, and eventual navigation antennas. Only the vehicle configuration part of LD was considered for this project, and not the route drawing.

Part of the vehicle configuration of LD is done when initiating a new system, such as what type of vehicles will be used in terms of wheel configuration, their wheelbases, and physical measurements (see Figure 2). When the system has been initiated, additional settings can be made to the vehicles that were added initially, and additional vehicles can be imported from other systems. Here the antennas can be placed (see Figure 3). The reason antennas are placed, but not other navigation equipment such as reflector scanners is that these other methods do not affect the path drawing.

There is also the possibility to place safety sensors and define their fields (see Figure 4). This is copied straight from the sensor software. It is preferred to import the settings, but this is not always possible. The safety cases are defined as well (see Figure 5). The cases define during which circumstances which safety field should be used.

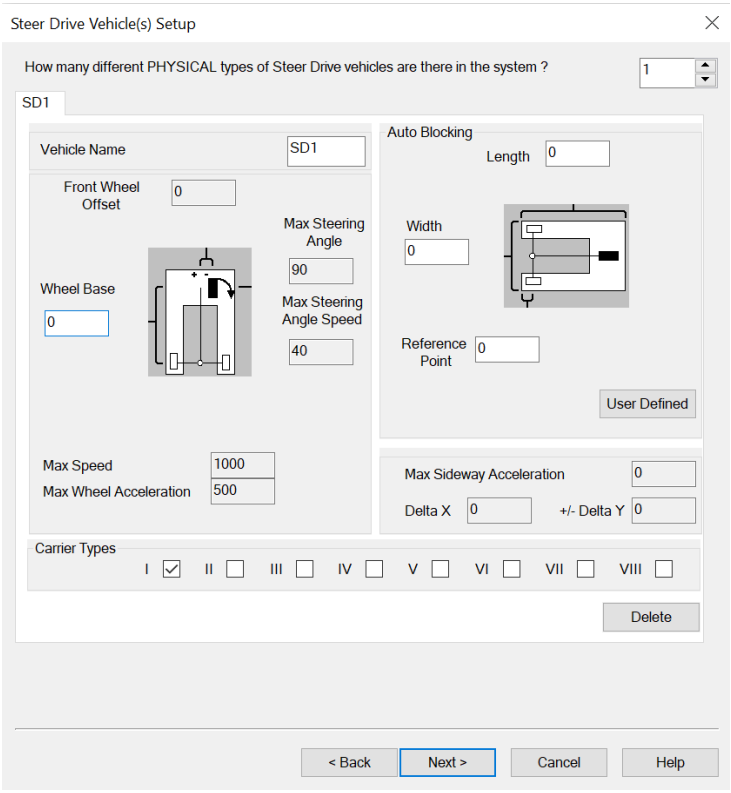

Figure 2: Layout Designer initial setup.

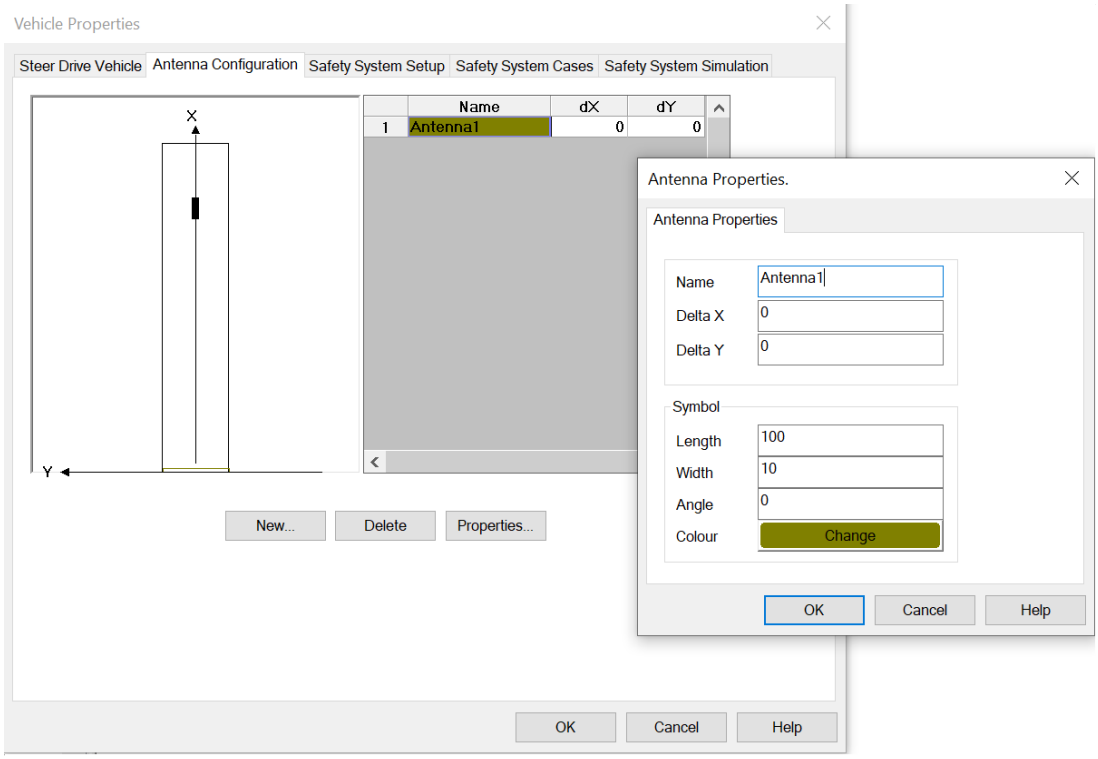

Figure 3: Layout Designer antenna setup.

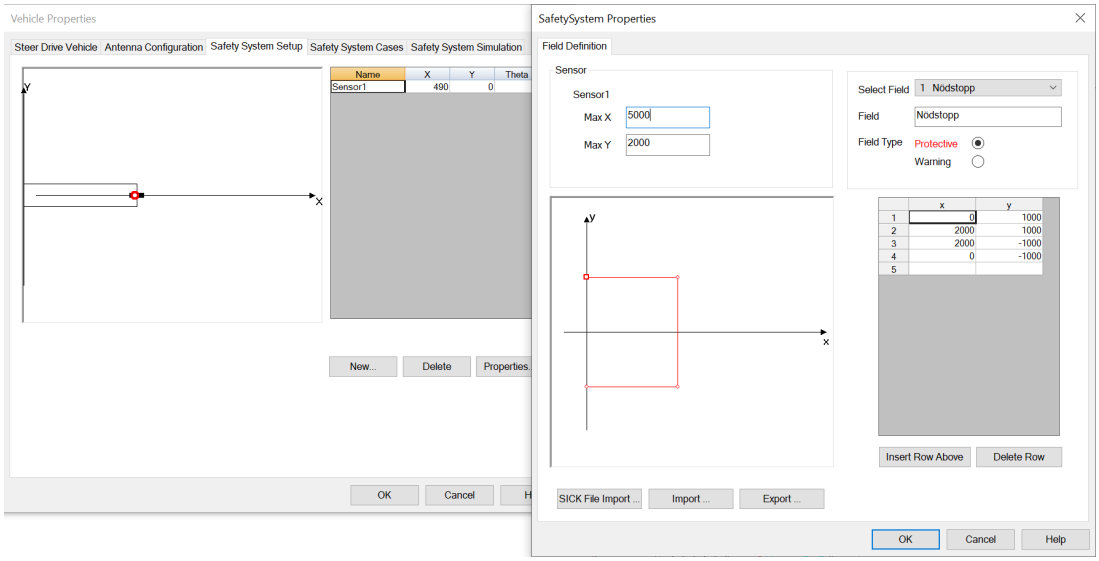

Figure 4: Layout Designer safety sensor and field setup.

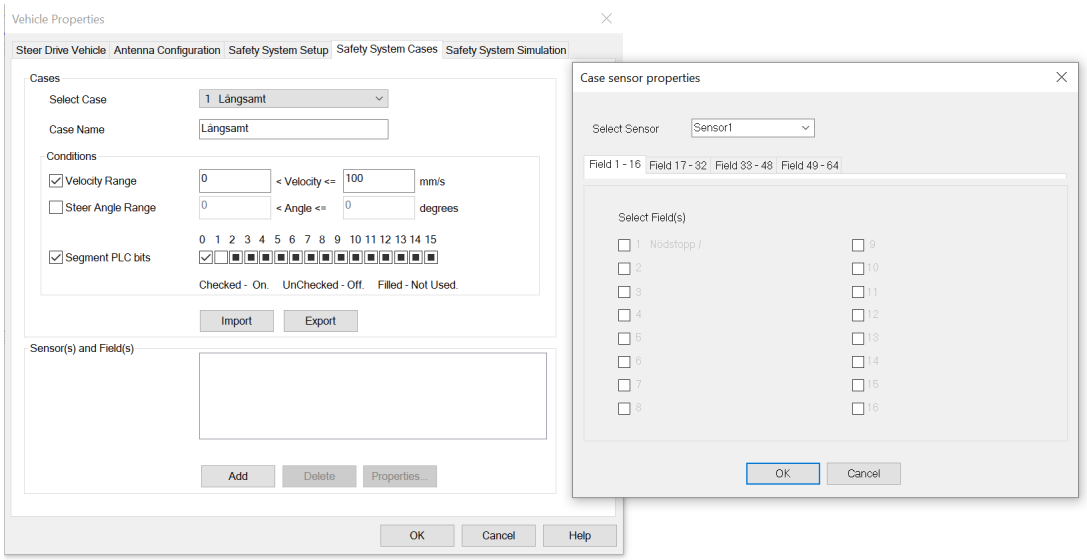

Figure 5: Layout Designer safety case setup.

Reflector Surveyor

To create a map of a site, a survey must be made with the present navigation method. The survey is done by driving around with an AGV and using different surveyor tools depending on the navigation method. When preparing for reflector navigation, the software Reflector Surveyor is used. This features a setup of the vehicle to be used, although its parameters can most often be imported from the connected vehicle. For the cases this is not possible, the setup is made as can be seen in Figure 6. In the figure, an image is shown over the vehicle with components such as wheelbase and scanner. The image is a mere representation so that the user will know what the position parameters mean and is a static image.

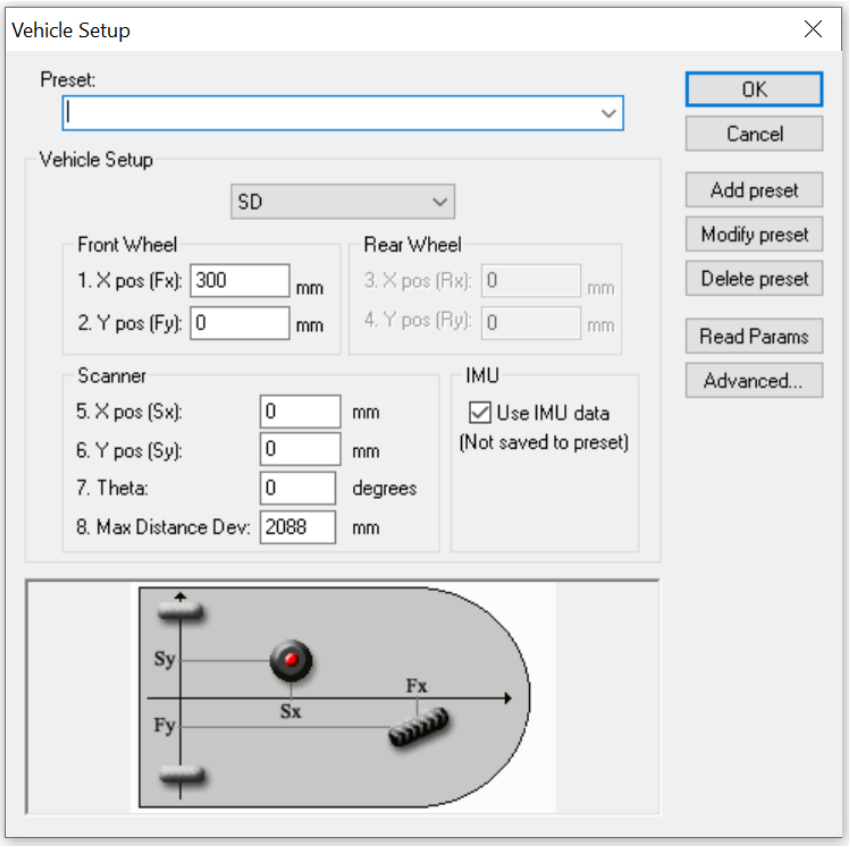

Figure 6: Reflector Surveyor vehicle setup.

3. Theory

3.1. Design thinking

The process that this study followed can be described by the five stages of design thinking: **Empathize**, **Define**, **Ideate**, **Prototype**, **Test**. It is a methodology developed to help the designer understand the human needs in a problem and can help a designer to solve complex or badly formulated problems.

- **Empathize** is the phase where the designer gains knowledge about the user and the problem and their experiences and what motivates them. This can be done by, for example, consulting experts through interviews.
- The **Define** phase is when the designer analyzes and interprets the data found in the empathize phase. During this phase, the user needs and the problem description is defined. A clear image of the problem and the user is produced.
- During the **Ideate** phase, the designer starts the ideation of new ideas. By having performed the two phases prior to ideation, the designer is allowed to think freely but relevantly.
- During the **Prototype** phase, the designer builds a prototype of the ideas, or combinations of them, that have come up during the Ideate phase.
- Lastly, the designer tests out the prototypes in the **Test** phase. This can be done either together with users or not. The important thing is to test how well the prototype fulfills the user needs and the criteria for success defined.

Often, the steps can be in a different order and some of the steps, if not all, are revisited many times during the design process [5]. In this project, mainly the prototype and test phases were revisited multiple times.

3.2. Accessible web design

During the project, colour combination choices have been made in regard to accessibility so that it fulfills contrast recommendations for reduced eyesight. In order to do this, the Web content accessibility guidelines 2.0 (WCAG) published by World wide web consortium were used. Contrast choices are based on WCAGs recommendation for accessible web design, and it states that text on background must have a contrast ratio of at least 4.5:1. There are three levels of accessibility in web design; A, AA, and AAA, where A stands for the lowest qualifications and AAA for the highest. Usually, governmental websites and official documents are by AAA standards. Since the designed software is not directed towards the general public, AA is sufficient and is also in line with KMAs own ambition regarding accessibility [10].

3.3. User experience (UX)

ISO defines user experience or UX as a person's perceptions and responses from the use and anticipated use of a product, system, or service. This includes the emotions, beliefs, preferences, perceptions, physical and psychological responses, behaviours, and accomplishments that the user experience before, during, and after the use. The response is a consequence of

the brand image, presentation, functionality, system performance, and interactive behaviour of the product, system or service, and is hence wider than the term usability. Unlike the common misconception, usability is a part of the aspects that make up the whole user experience and not a synonym. UX is hence profoundly connected with feelings and the hedonic qualities of the user, and not merely functionality. Another common misconception is that UX only regards graphical interfaces and is limited to apps and websites; however, UX principles are meant to be applicable for all types of products and services. In this project, to capture the usability aspects, tests were conducted, and the usability heuristics were taken into consideration. To capture the more hedonic aspects of UX and the emotional response of UX, the Geneva emotion wheel was used and will be described [13].

3.4. Usability

Jakob Nielsen describes 10 principles for successful interaction design called heuristics, which have been considered during the project and design of the intermediate and final prototype. How the heuristics were used and implemented in the design is presented in section 8.2. The heuristics are presented and described below [1].

- **Visibility of system status** - Users should be informed of what is going on and have a reasonable amount of time to process that feedback. Therefore, feedback should be presented to the user preferably immediately.
- **Match between system and the real world** - The design should follow the language and order that the user is already familiar with.
- **User control and freedom** - If a user makes an action by mistake, the user needs to be able to exit or regret that action. Therefore, the exit options should always be clear.
- **Consistency and standards** - The words, situations, and actions should mean the same thing throughout the design and follow the standards of the entire platform or industry.
- **Error prevention** - The design should prevent the user from committing errors, or present a confirmation option before committing to an action.
- **Recognition rather than recall** - The user should not have to remember information from one view to another. The information that the user needs in different situations should be seen or easily retrieved.
- **Flexibility and efficiency of use** - The interaction should be able to be tailored to adapt to the way that expert users use the interface, but the expert way of working should remain hidden from novice users.
- **Aesthetic and minimalist design** - Interfaces should not contain information or elements that do not help the user; unnecessary elements must not distract from the necessary elements.
- **Help users recognize, diagnose and recover from errors** - Error messages should be clear and understandable and expressed in language and not error codes, as well as constructively suggest a solution.
- **Help and documentation** - Even if the solution is preferably self-explanatory, it is sensible to present help and documentation for the user. The help should be easily accessible and concise.

3.5. Usability testing

The usability tests were designed according to theory presented by Joseph S. Dumas and Janice C. Redish. In their book, Dumas and Redish support that working iteratively with testing and design is likely to lead to a useful design. The nature of the usability test can, and should, be different depending on context and the user, but generally, all usability tests share the following characteristics, according to the authors [14].

- **"The primary goal of the testing is to improve the usability of the product. For each test you also have more specific goals and concerns that you articulate when planning the test."** Furthermore, the authors state that the focus on improving the usability of the product is what largely separates a usability test from pure research.
- **"The participants in the test represent actual users."** The users partaking in the study need to be people who use the product today or might come to use the product in the future. It is further stated that the test must take into consideration not to have too experienced or inexperienced users, since in the first case, the user might not experience all of the usability problems that the product has, and in the latter, the test might lead to changes that are not, in fact, helpful for the real user.
- **"The participants do real tasks."** This means that before conducting the user test, it is important to actually understand the users and how they use a product. Furthermore, it is stated that in complex software that contain a large number of functionalities, a selection of the tasks should be made but should still be tasks that the user is likely to perform "naturally" and that have a likeliness to uncover usability problems.
- **"Observe and record what the users do and say."** The usability test consists of both observations of the users' actions and evaluation from the user. It provides a balance between opinions and actions.
- **"Analyze the data, diagnose the real problems and recommend changes to change those problems."** Both the qualitative and quantitative data are analyzed, and along with the observation used to diagnose the product's usability and recommend solutions that might solve the problems found.

Furthermore, according to the authors, at least two or three users should be included per subgroup and test to ensure that the behaviour studied is not idiosyncratic. This is further strengthened by the Nielsen/ Norman group that claims that having 5 users per test is sufficient to uncover most usability problems of that test. Nielsen/ Norman group further states that having at least 15 users in total is sufficient to uncover most usability problems of a software, but that dividing the number of participants over multiple tests is the most successful way to help improve the usability of the tested software. By iterating and testing each redesign, it is likely to lead the design to become more usable for the users [12].

3.6. Success rate

When trying to improve something, it is helpful to measure the results before and after. The area of usability is no different. With usability metrics, it is possible to track how the usability changes over time or compare different interfaces. However, measuring the usability in quantitative terms might interfere with gathering qualitative data such as user insights. A typical

usability metric is the time it takes for users to complete a task. However, to accurately time users, you cannot ask them to think aloud while doing the task, thus missing out on valuable insights. Therefore success rate metrics were used in this study.

The success rate is a usability metrics that show the "percentage of tasks that users complete correctly"[8]. With success rate, every successful task is rated 1, and every partially successful task is rated a fraction of 1, in this case 0.5. It is then possible to compute the success rate of each task by taking the average score of the participating users. For example, five users can be asked to complete a set of 6 predefined tasks. A user can either be granted 1 point for complete success or 0.5 points for partial success (partial success is also defined forehand). The success rate is then decided by the total score granted to all users divided by the total number of attempts. In the example above, let us say the total score of all six users was 10; the success rate would then be 10/6*5=0.33, a success rate of a total of 33 percent.

3.7. Geneva emotion wheel

Emotions can be difficult to evaluate, especially when it regards something that does not trigger much emotion, such as the usage of an interface. However, emotion and hedonic aspects are central in UX theory [13].

The Geneva Emotion Wheel (GEW) is a tool used for measuring emotions. It consists of 20 emotions, arranged from bottom to top from low control to high control, and from left to right by negative valence to positive valence [17], [16] (see Figure 7).

When using it to evaluate the emotional response of using an interface, participants are asked for every emotion of the wheel to what intensity they felt that emotion. They answer on a scale from zero to six, with zero being that they didn't feel the emotion at all, and six being that they felt it strongly.

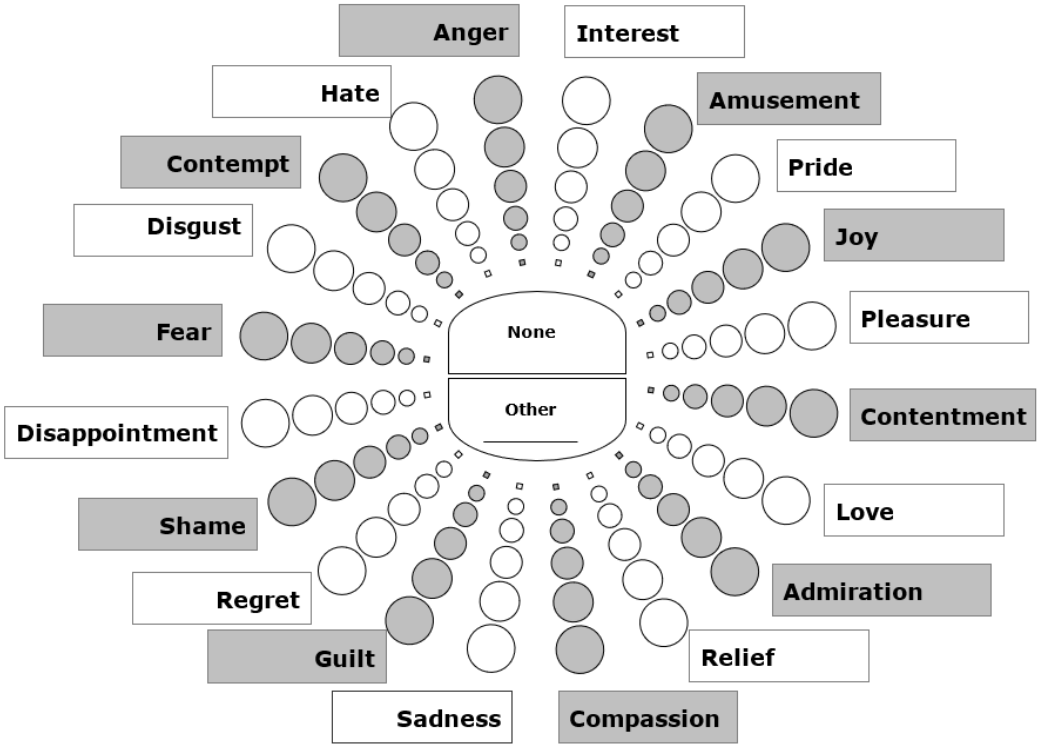

Figure 7: Geneva emotion wheel [17], [16]

4. Empathize

In this first step of the design thinking process, the users' ways of working, their needs, and how they use the current product were investigated. This was done by conducting interviews and observations with expert users and a usability test with success rate and an emotional response evaluation of the current KMA software with novice users.

4.1. Users

For the interview studies, in-house and partner application engineers were chosen as subjects since they are the primary users of the studied software. To be able to cover the needs of several different types of users, application engineers from three partner companies were interviewed (in addition to the interviews performed with application engineers at Kollmorgen). It was thought that these expert users would be able to tell us about issues that they had been experiencing with the software today and if they had come up with any strategies that resolved those issues.

The usability testing and success rate testing were performed with novice users since it was thought that they could give us further insight into how it can be to be a new user using the software for the first time. It would also be more just to compare the tests of KMA software to the new design since these users would have had as much experience with the current version as the redesign, which was not the case with expert users. The different users will be explained below, as well as some characteristics that were found in the study during the empathize phase.

4.1.1. Expert users

In-house application engineers
The application engineers' tasks at KMA vary, and they can act both as customer support and help partners set up systems at site. The idea of KMA software is, however, that the partner companies' own application engineers should be able to handle the software and set up the system for the end customer more or less independently. KMA does not manufacture AGV vehicles or sell to the end customer; the programs are seldom used by in-house application engineers to actually set up entire vehicles or formally set up entire AGV systems (the latter sometimes occurs when partner companies need much help). The application engineers at KMA have extensive knowledge about partnering companies, the ways in which they are working and the software they use.

Partner company users
For partner application engineers, their primary tasks are to adapt their products to their customers' needs, such as drawing the paths the AVGs will eventually drive in the warehouses and recommending suitable vehicle models. In some cases, they also adapt the vehicles to customers wishes, whereas in other cases, partner companies work with set models. During the studies, interviews were conducted with both in-house application engineers and representatives from three partner companies: Company 1, Company 2 and Company 3. The three companies differ in that Company 1 does not make customizable vehicles, Company 2 makes somewhat customizable vehicles, and Company 3 only manufactures very niche and customized vehicles.

Company 1 is a major global supplier of AGV and AGV systems and has a wide range of

set models. This means that Company 1 does not usually adapt their vehicle to the customer needs, and the environments in which the AGV models are meant to work in are also set. This means that vehicle configuration should only have to be done when a new vehicle is being developed, which is not very often, and when updates are made on the existing vehicles. However, the software that handles drawing the paths of the AGVs (Layout Designer) also requires the user to do some vehicle configuration. Company 1 has come around this by implementing so-called templates, already configured vehicles that the user imports into the different software. Since Company 1 is a large company, handling the layout for a site (in Layout Designer) and setting up a new vehicle template, or changing something in the vehicle templates are conducted by different people. Company 1 has a low need for flexibility in terms of adapting software to tailor-made vehicles.

Company 2 manufactures a wider range of different types of AGVs and can also customize different vehicles to their customers' wishes. This means that they both work with templates (already configured vehicles) and regularly set up entirely new vehicles. Working this way means a more frequent vehicle configuration, and users spend much time using Vehicle Application Designer. Company 2 have a mid-range need for flexibility. Often when constructing a custom vehicle, a user at company 2 will look back at previous vehicles that they have worked on in the past, and consider what parts of the configuration that can be put into the current vehicle project. This means that a user from company 2 relies much on experience working on similar projects and their own "library" of vehicles that they have built up over time. The users in Company 2 do not either have a set way of sharing previous projects with each other. This leads to new users at Company 2 often having issues in knowing what to do.

Company 3 only custom makes vehicles, and this means that Company 3 do not use vehicle templates. At Company 3, there are "as many different models of vehicles as there are customers". Their vehicles are primarily directed towards industry, but they also have experience in manufacturing, for example, autonomous vehicles for clinical environments. This means that they work much with vehicle configuration, and also have a high need for flexibility, and have also come up with their own workarounds allowing them to push the boundaries of the software's ability to be flexible. However, even if all of their vehicles look and function differently, they take parts from old configurations when working on new projects, as Company 2 does.

4.1.2. Novice users

During the Empathize phase and testing of the current KMA software, the number of novice users was five in total, plus three pilot test users. The group of users consisted of both KMA employees (3) who had not been in contact with the software before and non-employees (2). The non-employees were people with various technical backgrounds such as electrical engineering, IT, design engineering and automation and mechatronics. They were in many ways used to handle complex technical interfaces. Both user groups were thought to represent a user who might be new in their position and opens the software for the first time. It was expected that the two different user groups would get different results and perhaps that the group consisting of KMA employees would do considerably better on the usability tests since they were more familiar with the terminology used at KMA. However, the two groups got very similar results on both the tests performed with the old software and the test performed with our prototype.

4.2. Interviews and observations with expert users

The interviews were carried out remotely via Microsoft Teams with expert users. The interviewees were in-house application engineers at KMA (3 in total) and three application engineers from the three partner companies. The interviews were recorded and transcribed. The interviews were changed and were tailored for each type of user in order to get as much relevant data from each interview. Prior to conducting the interviews, a loosely formed user journey had been constructed in dialogue with KMA employees. This user journey was then continuously formed together with users throughout the interviews and was used as a mediating tool. The interviews and the iterated user journey were conducted to investigate the user needs expert users had.

The interviews were of semi-structured nature where the researchers could both ask specific questions that were decided beforehand, questions that arose in the moment and quick observations in relevant software. During the interviews in this part of the project, Miro was used to construct a schematic user journey with the user. The board was later developed and evaluated with the next users interviewed. During the interviews, the participants were asked and encouraged to open the software that were discussed and share their screen. This was done in order to use the software as mediating objects and to let the researchers observe how the users worked with the software.

Using the iterated user journey as a probe provided insights into the ways that the users were working. The users described their process of setting up a new AVG system and configuring vehicles, from getting the first inquiry from a customer to setting up the system at the site. The questions, both the predetermined and the questions that arose in the moment, together with the usage of the KMA software as mediating objects, provided insights on what parts of the process were the most challenging and what strategies the users performed in order to overcome those challenges.

In the case of studying expert users, interviews and observations were also favoured before usability testing of the current KMA software since it was reasoned that they were likely to have discovered and learned to avoid most mistakes caused by poor usability and that their struggles were likely to be better described in an interview. The choice was also based on the fact that interviews were possible to perform remotely and preferred over a survey also since it often provides researchers with deeper knowledge. The interviews with the expert users were transcribed and analyzed using a KJ method.

4.2.1. Key insights from interviews with expert users

Some of the main themes that experts brought up during the interviews are presented below:

Users

- Generally, different people work with the Layout Designer and Vehicle Application Designer
- The KMA software are difficult to learn for novice users. One user suggested that "The program could include tips for the user on what to do next"

Import and export

- Expert users find it time-consuming that they are not able to import one vehicle at a time into LD
- Importing and exporting different types of files to and from the different KMA software is time-consuming
- Being able to import components from manufacturers is useful, such as importing a safety scanner or separate antenna.

Templates and flexibility

- The solution must be able to be used with both custom vehicles and standard vehicles (both for template and non-template vehicles)
- Expert users who use templates find it time-consuming not being able to update multiple templates at the same time
- Users rarely start from scratch when configuring a new vehicle, but use parts from old vehicles. Parts from up to 5-6 vehicles can be used.
- Some users copy and paste from the HTML code of VAD. It is time-consuming to copy and paste parameters. This is why it is "shortcut-ed" using HTML. Drag and drop would be preferred.
- Adapting what is being presented to the user depending on what type of vehicle is being configured would be a good idea.
- There is a need to be able to divide different vehicles into subgroups both regarding physical qualities and behavioural qualities

Errors and documentation

- If an error occurs, it needs to be easily detected since users are often working with many parameters at the same time, slips occur
- It is not recommended to tamper with the templates since there is no way to document what has been changed
- Documentation about the vehicle is needed, users usually write in a Word document to communicate changes that have been made

Other

- For VAD, there is no need for shared usage today, but for LD, there might be
- The software are in need of a modernization regarding aesthetic qualities

4.3. Usability test with novice users

A usability test was made in order to map out the most common use errors and obvious usability flaws in the software conducted by novice users, and why the users had difficulties in understanding parts of the program. The success rate of the users was measured for later comparison with the final design. The usability tests were conducted remotely via Microsoft teams, where the participants got to remotely control a computer with all of the programs tested installed. At the end of the test, the users got to reflect on the usage. They were continuously encouraged to think aloud when performing the tasks to obtain some qualitative data. All tests were recorded and used for analysis.

The participants were asked to perform tasks that had been previously designed to test parts of the programs. The tasks could be described as sub-tasks expert users would perform when setting up a new AGV system. It could be, for example, to import a vehicle from a previous project or to configure a vehicle with certain parameters. The test consisted of 20 tasks with some sub-tasks. The success rate of the users was measured for later comparison (see Appendix A).

Throughout the tests, the users kept the researchers informed about their thought process by verbally explaining how they guessed a certain situation should be handled or other things that they thought of during the test. After the test was conducted, the participants were asked to reflect on the usage and the emotions that the usage raised. The comments and the reflections were also analysed using a KJ method, and the usability of the tested software was evaluated using success rate.

4.3.1. Emotional response

An evaluation of the novice users' emotional response to using the software was conducted. It was reasoned that it could provide more insight into the users' holistic impression of the software, in addition to evaluating mere usability. In order to help users identify their emotions, a Geneva emotion wheel was used as a basis for reflection after each usability test with novice users.

The participants were asked to reflect upon how the usage of the interface made them feel, and to what level of intensity. They were also asked to motivate why they would rate the intensity of the feeling in a certain way, although it was made clear that they did not have to motivate the emotion if they could not find a motivation.

The most prominent emotion users felt was **interest**, with an average of 3.7, followed by **pride**, **relief** and **anger**, all with an average of 2.3.

Positive response
Some of the comments regarding positive emotions, or lack of negative emotions, were:

- "This old interface looked pretty advanced, so I felt a bit proud when I got it right"
- "When I found what I was looking for, I felt content"
- "You can change everything afterwards. If you couldn't, I would have felt more regret"

Negative response
On the other hand, some comments about negative emotions, or lack of positive emotions, were:

- "If I would have gotten a confirmation that I had done the right thing, I would have felt very happy, but I didn't. So now I'm left with a feeling of insecurity."
- "Getting more feedback would have been good in order to feel more relief ... Now I'm not sure whether it will work"
- "I understand that this is done by engineers, for engineers"
- "It made me a bit sad how inconsistent things were"
- "It didn't feel shameful to make a mistake, it felt like reasonable things to get wrong"
- "It's difficult to understand what you're supposed to do, it makes you feel stupid"
- "I feel technology anxiety, anxious to start messing with the system"

4.3.2. Usability issues

Based on the usability tests with novice users, some of the usability issues that were found throughout the three software are presented below:

Visualizations

- The icons used in the software are difficult to interpret for novice users
- The vehicle representation images used to represent the vehicles in the software are difficult to interpret for novice users; the image used in Reflector Surveyor is easier to interpret than the one used in Layout Designer
- Novice users expect vehicle representation images to be updated depending on the input perimeters
- Novice users expect to be able to interact with the vehicle representation images. In the current software, the visualizations are a mere representation
- The antenna representation is not visible
- In RS, the vehicle representation image does not correspond to the default values that the program has when opening it, which leads to confusion

Feedback

- Novice users, in general, feel insecure if what they have done is correct
- Novice users are in need of visual feedback that their inputs have been executed

Import and export

- It is difficult to import a vehicle
- It is not possible to only import one vehicle in LD; the user has to import all vehicles within a drawing
- It is difficult to see where the imported vehicles end up

Consistency

- Layout designer vehicle configuration behaves inconsistent. For example, for some parameters, a dialog box opens up, and for some, the user is expected to input parameters in the main view
- Short commandos such as tab are not available

Navigation

- Novice users find it difficult to navigate within the vehicle configuration window and forget where different parameters are within the window if they need to go back
- It is difficult to find the safety system cases
- Settings are not where the user expects them to be
- Users mistake the "next" button for creating a new vehicle
- Users feel insecure if "next" is the same thing as "save" and if they are able to go back if they press "next"

Other

- The PLC check-box setting in LD is difficult to understand
- The roman letter system over carrier types is confusing
- The software uses acronyms instead of typing out the entire names of things which makes it difficult for novice users to find things

5. Define

In this step, the data gathered from the Empathize step was organized with a KJ analysis, a Hierarchical Task Analysis (HTA), an impact map, a function tree, and a user journey map. These were then translated into user needs.

5.1. KJ analysis

The KJ analysis, also known as the Affinity diagram, is a method used for structuring raw data. It is useful for grouping data to enable solving problems on a higher level, rather than addressing every issue separately [2]. It is usually done by writing every data on a post-it note, organizing the notes into groups, naming the groups, and prioritizing them.

To organize the raw data from the interviews and the usability tests, a KJ analysis was carried out. This method was chosen since it is suitable for organizing qualitative, unstructured data, such as the recordings from the semi-structured interviews with expert users. The recordings of the usability tests with beginner users also contained a lot of comments and user behaviour not included in the actual test. Therefore the KJ method was also chosen to analyze the usability test.

Due to the restrictions of physical meetings during the Covid-19 pandemic, the analysis was done digitally and not with physical post-it notes. The software Miro, which is an online software for visual collaboration, was used.

One analysis was conducted for the three in-house application engineers at KMA, and three for each of the partner companies. This was done to be able to separate the needs of different user segments. For the usability test with the beginner users, one analysis was made.

The first step of the analysis was to gather interesting quotes from the interviews and the usability tests. These were things like indications of how the users use the software, needs that are not fulfilled by the current software, what the expected behaviour is, or anything that is unclear.

The quotes were written onto digital post-it notes in Miro. These were then sorted into groups of notes with similar contents. After the groups were formed, they were named based on the contents of the notes. In some cases, large groups were divided further into smaller groups.

5.2. Hierarchical task analysis

The hierarchical task analysis (HTA) was conducted to help understand in what order the user performed certain tasks within the program. An HTA was performed on Layout designer (see Figure 8), and Vehicle application designer (see Figure 9). The results were to be used as a guide to understanding how the hierarchy of the different functions should be placed in a future design concept.

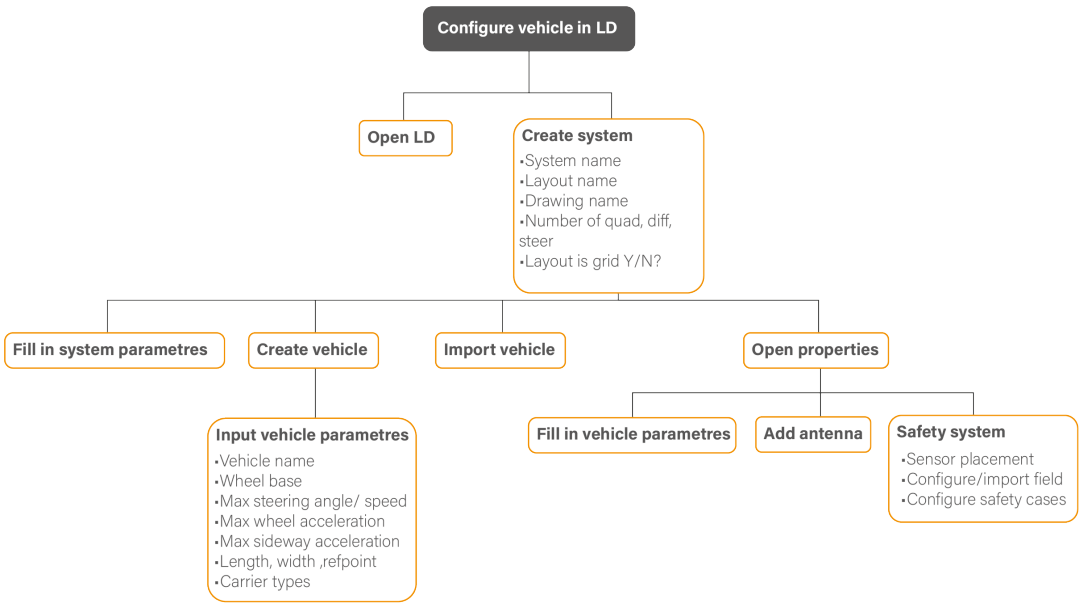

Figure 8: Hierarchical task analysis for Layout Designer.

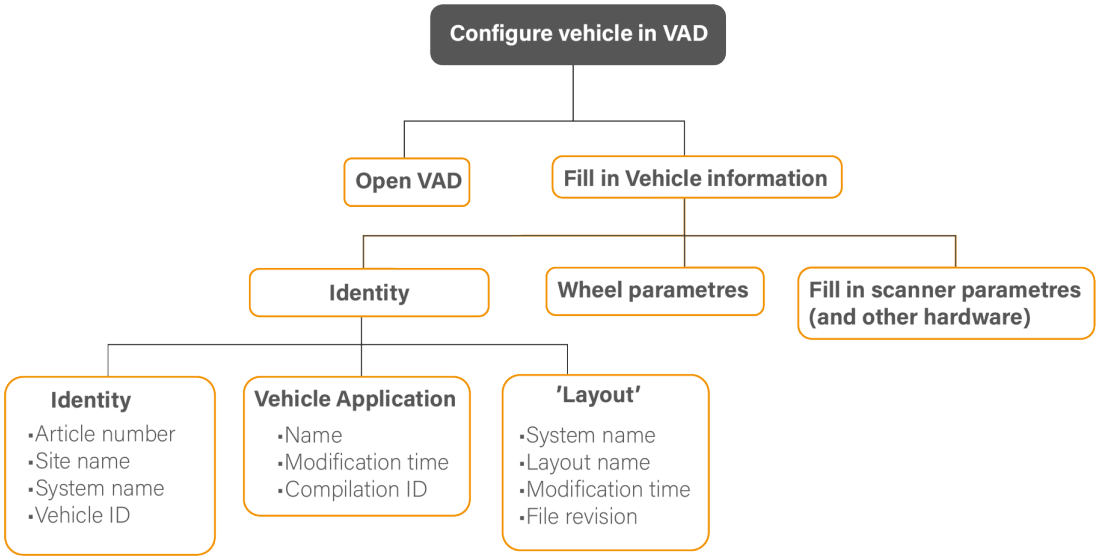

Figure 9: Hierarchical task analysis for Vehicle Application Designer.

5.3. Impact map

The process of impact mapping is used for connecting features, or deliverables, to the overall goal of the project. It can be done to identify needs, and how to solve those needs, by presenting some form of deliverable [11]. In this project, it was used to create a map of how the different user needs would impact the product, and in turn, its overall goal. This way, the impacts could be prioritized to help in the process of prioritizing the user needs.

The impact map was done by first defining the overall goal of the project. Then the actors related to this goal were defined. After this, the impacts on each actor were identified, and user needs were correlated to the impacts. The resulting impact map had "Improve customer satisfaction (of the product)" stated as the project goal. The actors were defined as the different types of users, KMA application engineers, partner company application engineers making custom vehicles, ditto making standard vehicles, beginner users, and service technicians. Seventeen impacts were found, factors impacting the actors' use of the product (see Figure 10).

Since the overall goal of the project was defined as "Increase customer satisfaction" of the product, the impacts were closely related to usability and could be seen as primary user needs. The actors, in this case, were the different segments of users.

5.4. Function tree

A function tree is a tool for functional analysis. It is a map of how all functions in a system are related to each other. It includes one main function, and any number of sub-functions [18]. It is useful for defining what functions are needed or wanted in a product, without assuming what solutions are used. The current interfaces were used as a base when constructing the function tree.

The main function was defined as "Configure a vehicle". Primary sub-functions were identified, such as "Input information". Auxiliary functions were also taken in regard, functions that help the user but are not directly a sub-function of the main task. This was for example "Visualize input". One level down the functions was broken down further, and at the next and lowest level were possible solutions to the sub-functions. For example, the primary function "Guide user" is broken down into the secondary functions "Describe fields", "Explain functions", and "Avoid complexity". At the next level, "Explain functions" has the possible solutions '"Help"-function', "Clear symbols" (see Figure 12).

The possible solutions were included in the Ideate-phase. Since it was already decided that the resulting product would be software meant to be used on a PC, the possible solutions were all applicable to this criteria.

5.5. User journey

A user journey was constructed throughout the Empathize phase of the project. The user journey was based on findings from the interviews and consisted of the whole process of developing and implementing a new AGV system at a customer site. It provided a structured view of what the most important functions and software were for the user in different stages of the development and implementation of a new system and placed the user in a context.

The user journey starts with the construction of the vehicle. For the application engineer, this means working in VAD to set drives, wheel configuration, and sensors. Usually, the user starts with an old configuration that is similar to the new vehicle, and makes modifications to it.

They receive the site blueprints from the customer, and after this start preparing the routes in LD. When doing this, they also make vehicle configurations to be able to draw routes correctly. This includes entering sensor data, which is available in the sensor software that is separate from the KMA software. Sometimes it is possible to import it directly, other times it must be copied manually. After the layout is created, a capacity simulation is performed to see how many vehicles will be needed at the site. This can be done in iterations while having a conversation with the end customer. After the capacity is established, the system is sold.

In the case of reflector navigation, the next step is to make a suggestion for reflector positioning. This often includes site visits. When it has been decided where the reflectors should be placed, the end customer mounts the reflectors. A site visit is then made to scan the reflectors to get their actual exact positioning. When doing this, the software Reflector Surveyor is used. This is connected to the vehicle and in most cases receives the vehicle information automatically. For older vehicle types it might however be necessary to enter this information manually.

Trimming of the vehicle is also done, usually during the same site visit. A brake test is done to see how long the stopping distance is and adjusting the safety system accordingly. The vehicle is also calibrated and these parameters are set in VAD.

After this the layout may be modified with the reflector positioning, a test drive is made and a final flow and capacity simulation. After this, the system goes live, and the tasks of the application engineer change, and the role becomes more of a supporting role with troubleshooting errors and maintaining the system.

5.6. User needs

A user needs list is a list of all the needs of the user that have been identified through previous user research. It is useful as a base for ideation, to find solutions to each separate need. It is also useful as a means to evaluate concepts later on in the process, by seeing how well the concepts fulfill the most important user needs.

The user needs list was successively built and edited during the Empathize and Define stages. In the list, each need was written down, along with a category such as "Clarity", "Collaboration" or "Flexibility", and an indication of whether the need should be seen as a guideline or requirement. Guidelines, in this case, are more general about the desired behaviour of the solution, for example, "Visualizations should be clear and serve a purpose to the situation where they are presented", and "Allow sharing and re-use of configurations". On the other hand, requirements are a stricter description about what functionality should be included, such as "The sensor locations can be configured". The eight most important guidelines were selected and condensed down from the complete list of guidelines that are presented in Appendix C. The following eight guidelines were also the basis for the guidelines presented later in Chapter 8.

- The most important information should be the most visible

- Images and visualizations should be clear and serve a purpose to the given situation in which they are presented
- The user should receive feedback when making an input
- Allow sharing and reuse of configurations
- Allow for communication between KMA, customer ,and end-user
- The solution should behave in a consistent way throughout usage
- Ease usage for beginner users
- Allow for iterative work
- Allow for easier copy/ paste options

There were also requirements set about the functionality the solution should contain. Although there might be more parameters to be added, the work was based on a few central functions. These were selected as follows:

- Allow the user to divide vehicles into different groups
- The wheel configuration and wheel positions should be set
- Means of navigation, such as antennas or reflector scanner, can be configured
- The sensor locations can be configured
- The safety fields can be configured but also imported

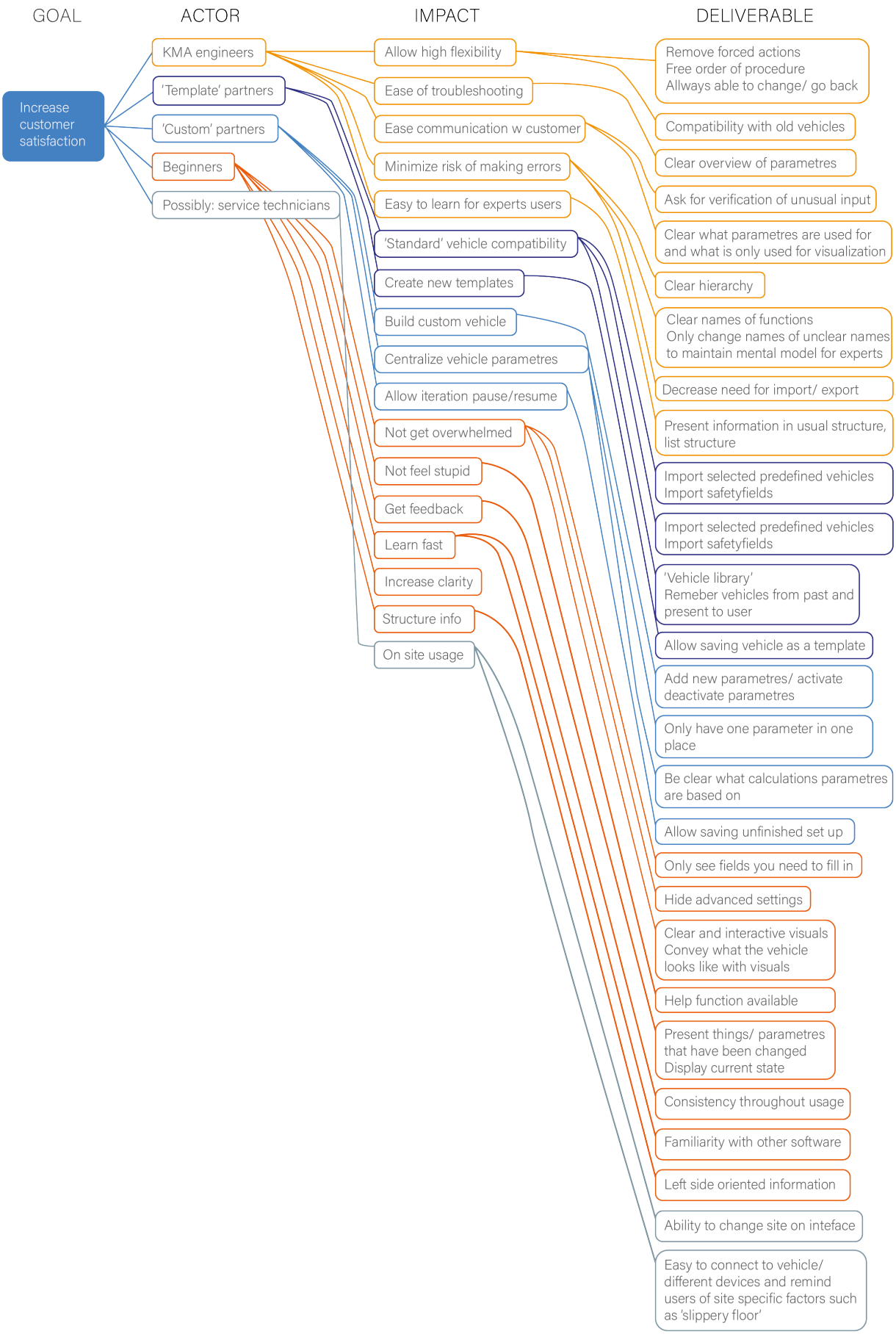

Figure 10: Impact map of the goal to increase customer satisfaction.

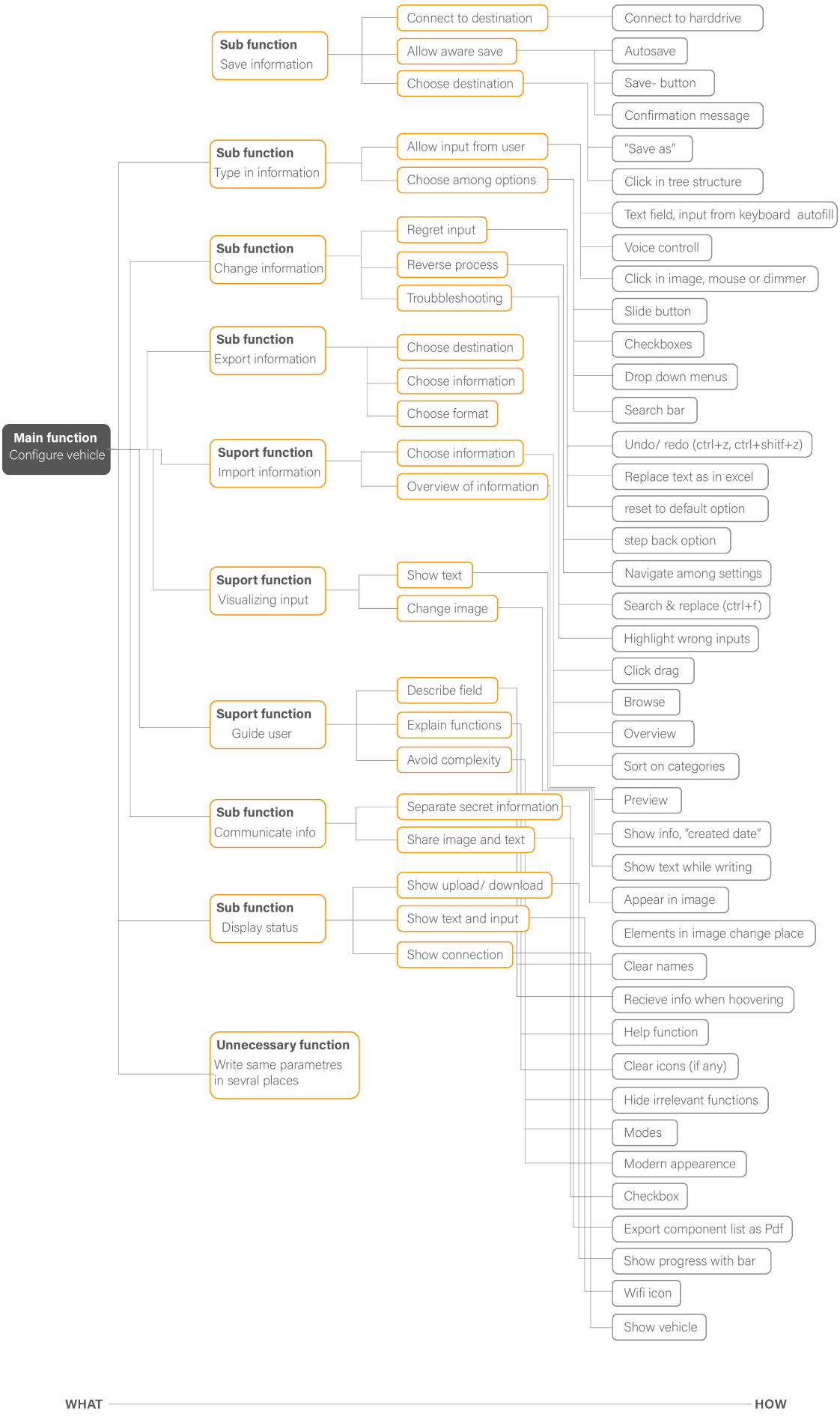

Figure 11: Function tree of the function to make a vehicle configuration.

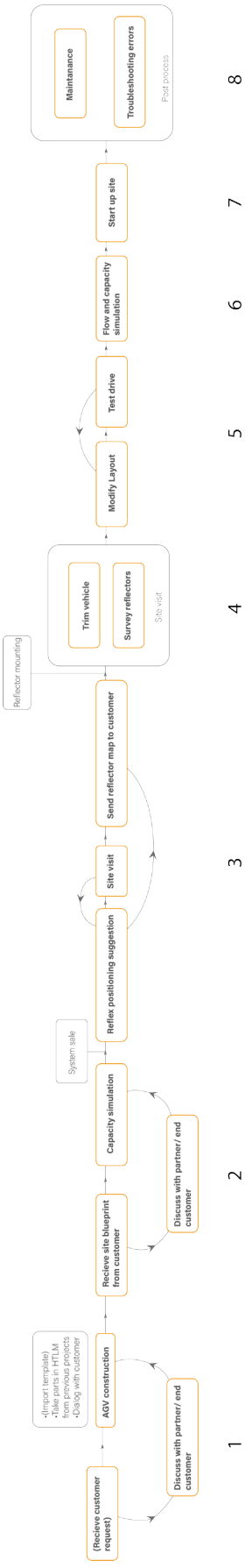

Figure 12: User journey of the process of setting up a new AGV system.

6. Ideate

6.1. Brainwriting 6-3-5

To start generating ideas for solutions, the method of brainwriting 6-3-5 was used. This is a tool used for generating ideas [18]. With this method, the ideas of one participant is built upon by another. A set of themes or questions are decided upon; these are then written on the top of a paper. Each participant has a specified amount of time to write as many ideas as possible. Then they give it to the next participant, who adds more ideas and builds upon the previous ideas written on the paper.

The method was only used among the two researchers. The themes were selected beforehand and were based upon some of the central findings of the "Define"-phase. The themes were both specific functionalities that were to be developed, such as "Safety fields" and "Physical parameters", but also included some more abstract themes such as "Learn fast" and "Flexibility". The themes that were used were the following:

- Prevent errors - How can errors be avoided or mitigated?
- Templates - How can the use of templates be facilitated?
- Flexibility - How can the flexibility be increased?
- Learn fast - How can the solution be easy to learn?
- Structure - What structure should the solution have?
- Safety fields - How can safety fields be configured?
- Physical parameters - How can physical parameters be configured?

The themes were divided between the two researchers who had 5 minutes to write down and sketch their ideas. They then switched papers and built upon each other's ideas. This procedure was repeated twice so that each person had every paper twice. The session resulted in a large number of ideas. The results were discussed after the completion of the session and used for developing two concepts in Figma, called **Matlab** (see Figure 13) and **Tree structure** (see Figure 14).

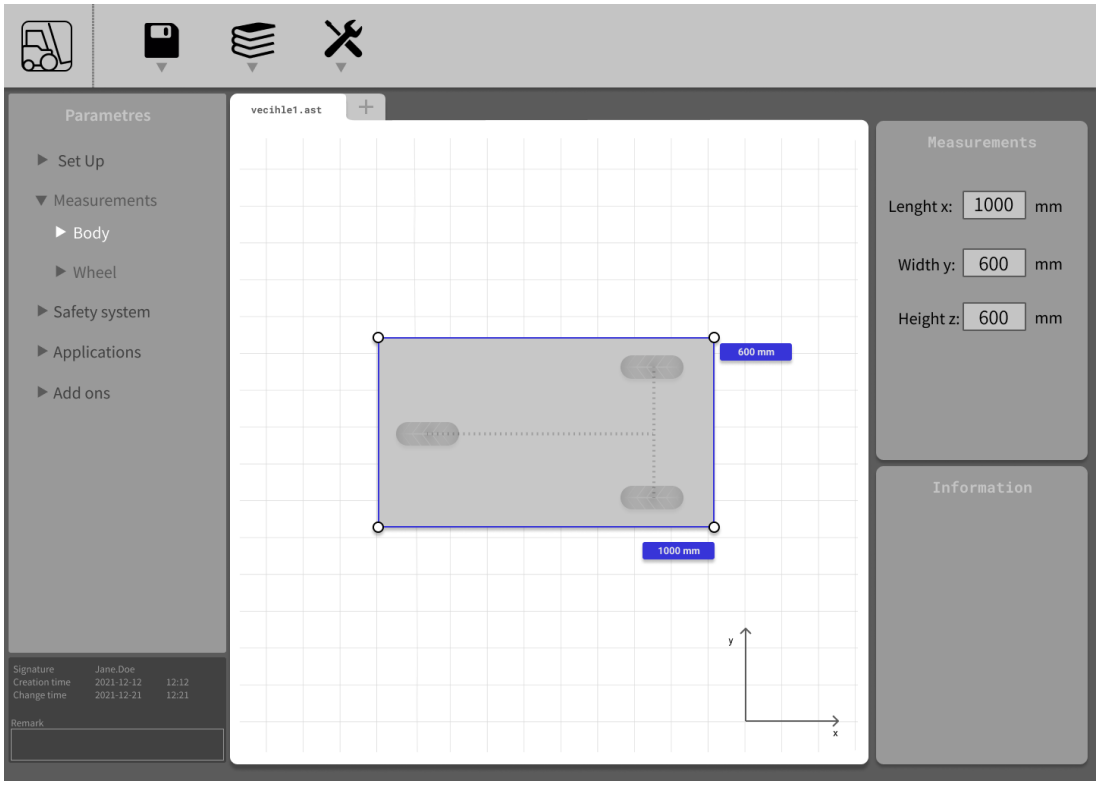

Figure 13: Early concept "Matlab".

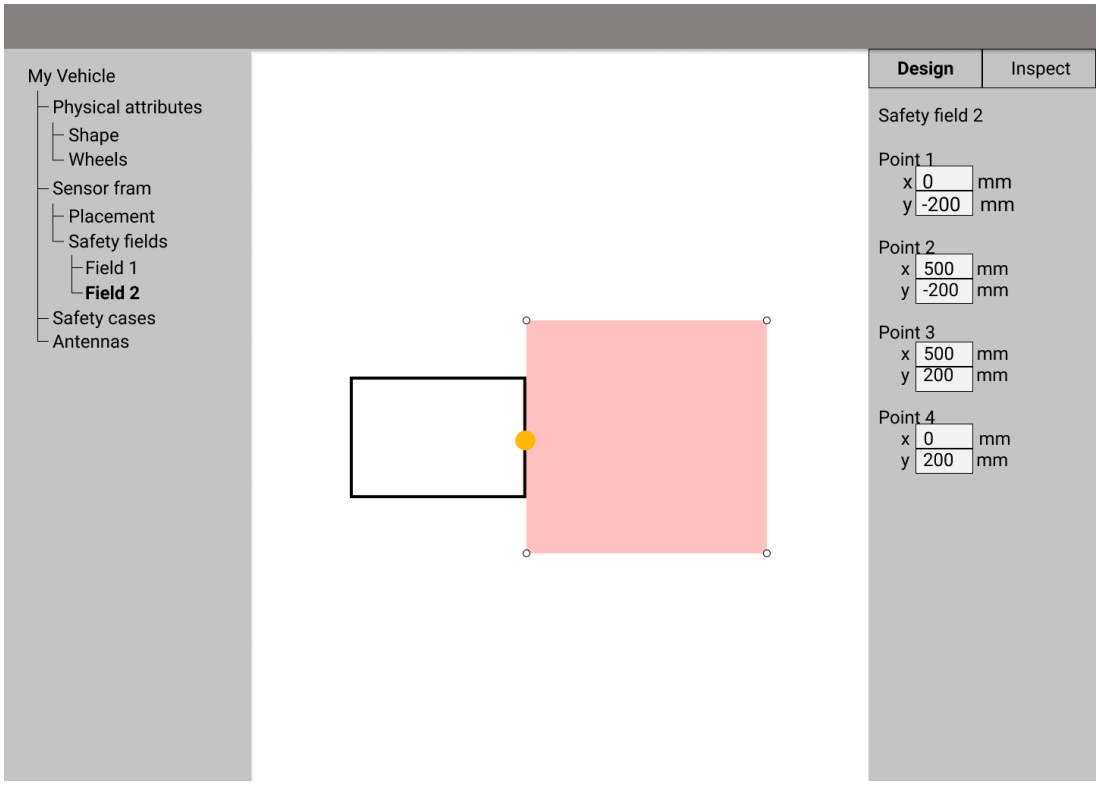

Figure 14: Early concept "Tree Structure".

6.2. Brainstorming with random words

To keep exploring the solution space, and to avoid choosing the first concepts that were developed, a brainstorming session was conducted. The goal of the method was to generate as many different ideas that could be used in the new software [18]. Prior to the session, themes were picked. The themes were constructed using the list of user needs and consisted of:

- How can the vehicle and its parts be represented?
- How can this product be incorporated in a bigger system?
- How can users be enabled to work with "standard vehicles"?
- How can the user be reminded of the real-world system?
- How can the user be encouraged to work in iterations?
- How can communication between users be made easy?

Since the brainstorming session was only conducted by the two researchers, a facilitator of the session was lacking. This was solved by adding a random words generator as a structure to the session. The researchers wrote as many ideas as possible, using a random word as a probe or mean for inspiration.

As in all brainstorming sessions, the researchers made sure not to be critical about the ideas, and focus on quantity rather than quality. The ideas were sketched or written on post-it notes and placed visibly during the entire session so that new ideas could be formed from previous ideas. The ideas were formed individually and in silence, but explained when placed among the other post-its. The session resulted in around 150 different ideas, many outside-the-box but also many feasible ideas. Some of the ideas that were later used in some form included:

- Every software in the system is a card on a dashboard
- The vehicles have a short description, like the backside of a book
- It is possible to merge two templates
- Every change is signed by the user, so it is clear who to ask about it
- Receive search suggestions based on previous searches
- Add objects to the vehicle by click-dragging them
- The vehicle representation is built by Lego-style blocks
- A view that is a model of the site where you can click around
- Receive notifications when the configuration needs to be updated due to changes at the site

6.3. Morphological matrix

A morphological matrix was used for combining different ideas into entire concepts. It was done by assigning key attributes to the columns and putting ideas for each of them in the rows [4]. In this case, the themes of the prior brainstorming sessions were assigned as the attributes in the columns. The ideas generated by the brainstorming sessions were inputted into the rows. Since the brainstorming sessions were very free, sometimes the ideas generated fit better into one of the other themes or even a new one. In these cases, the ideas were moved around, and some new attributes were assigned. The morphological matrix in its entirety can be seen in Appendix B.

The elements from the Morphological matrix were combined in different ways to form complete concepts. Five concepts were formed. Three in which the elements were chosen intentionally, two in which the elements were chosen at random. These concepts were developed further and sketched on paper. The sketches showed different functions from the morphological matrix. The following concepts were generated with the morphological matrix:

- **Social media** - An online collaborative concept focused around a 3d model of the vehicle. The settings are made in a dialog box, and there is a menu at the bottom of the screen which shows what step of configuration you are at and which you have completed (see Figure 15).
- **Building blocks** - The vehicle in this concept is represented by building blocks that you can use to create any shape. The blocks are positions by click-dragging symbols, as are sensors and other objects you place (see Figure 16).
- **Point and click** - The navigation in this concept is done by clicking around in a model of the site. For example, when clicking on the sensor, you zoom in on it and can set its parameters (see Figure 17).
- **Layers** - A concept where the main view is defined by layers, such as a "Wheels"-layer, "Sensor"-layer etc, that can be checked or unchecked to be hidden or shown.
- **Reality slider** - A concept where the vehicle is represented by a model with a varying degree of "reality", from a schematic sketch to a photo, controlled by a slider. It features voice commands, and you can set parameters by email.

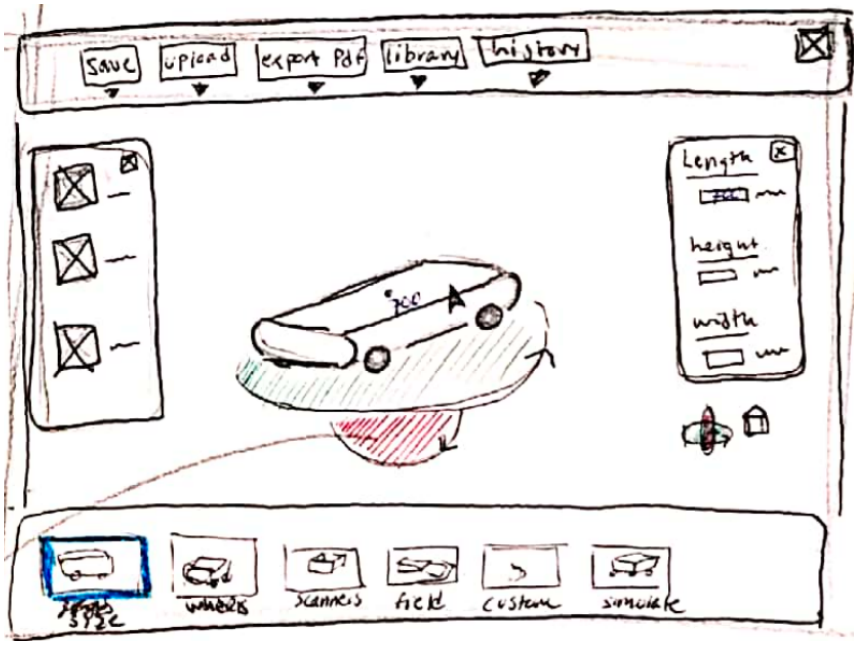

Figure 15: Generated concept "Social media".

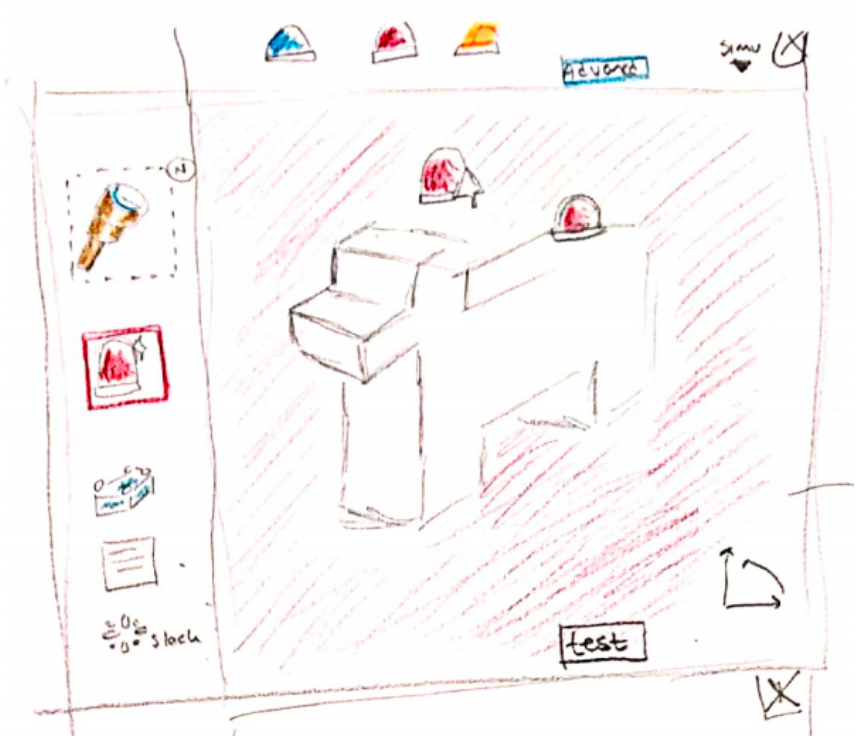

Figure 16: Generated concept "Building blocks".

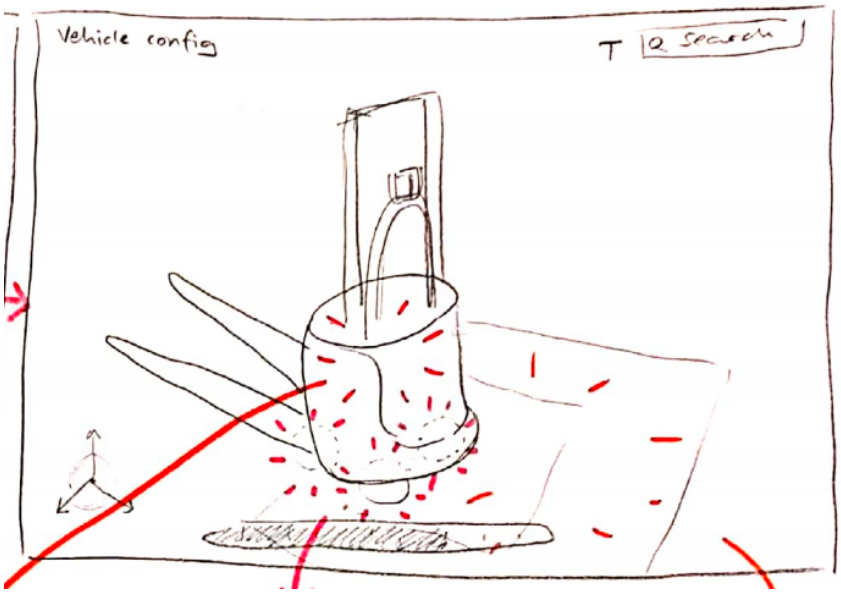

Figure 17: Generated concept "Point and click".

6.4. Pugh matrix

A Pugh matrix, also called a decision matrix, was used in order to rate the different concepts. When performing this method, a set of criteria is decided as a base for rating. One of the concepts is chosen as a reference and given a rating of 0 on all of the criteria. The other concepts are given a rating of +1 or -1 depending on if the compared concept performed better or worse than the reference on the criteria in question. The concept with the highest score is the concept most likely to fulfill the user needs [18].

The eight most important guidelines from the user needs list were used as criteria, as well as one requirement. The concepts that were compared were the two from the Brainwriting session, **Matlab** and **Tree structure**, and the five from the morphological matrix. The KMA software solution was also included. **Building blocks** was chosen as reference. The other concepts were compared to the reference and rated (see Figure 18). The winning concept of the Pugh matrix was the concept **Social media**, followed by **Tree structure**.

	Building blocks	**Reality slider**	**Point and click**	**Matlab**	**Layers**	**Social media**	**Tree structure**	**KMA softwares**
The most important information is most visible	0	+	-	+	+	+	+	-
Images and visualizations are clear and serve a purpose to the given situation	0	+	+	+	+	+	+	+
The user receives feedback when making an input	0	0	-	0	+	+	0	-
Allow sharing and reuse of configurations	0	+	-	+	-	+	+	+
Allow for communication between KMA, customer and end user	0	0	-	-	0	+	-	-
The solution behaves in a consistent way	0	-	-	0	0	0	+	-
Allow the user to divide vehicles into different groups	0	0	0	0	0	+	0	+
Ease usage for beginner users	0	0	0	-	-	-	-	-
Allow for iterative work	0	0	-	0	0	-	0	-
Sum		2	−4	1	1	4	2	−3
Rank	4	2	6	3	3	1	2	5

Figure 18: Pugh matrix

7. Prototype and Test

Prototypes were created and tested in several iterations. Research has found that using fewer users in each test, but doing more iterations, is more beneficial than doing fewer iterations with more users in every test [12]. Therefore the priority was to do as many iterations as possible, with 3 to 10 users in every test.

In this chapter, the process of how the concept was developed through iterating between prototyping and test. The prototypes were evaluated with beginners, experts, and UX experts through either usability testing or interviews.

7.1. Prototype 1

The concept most successful from the Pugh matrix, "Social media" was used as a base for a new prototype created in Figma. However, ideas from other concepts that were rated high were also included. This first prototype provided an overview of how the program would function, and explanations of some functions on a more detailed level than the sketches could. The prototype was constructed with a focus on functionality, rather than on graphic design and consisted of wire-frames with text in grayscale. This was done intentionally in order to first and foremost focus on the design of the functionality.

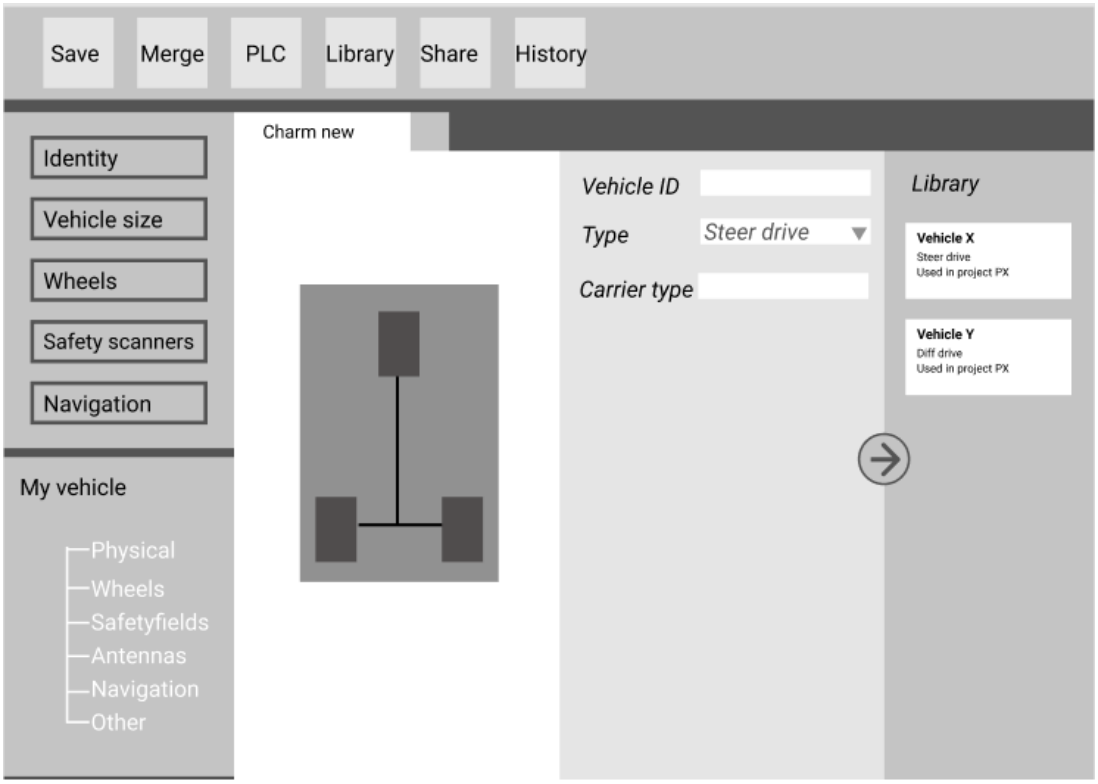

Figure 19: Prototype 1 identity view.

The prototype featured a split left menu with navigation buttons and an object tree with all elements added to the vehicle (see Figure 19). In the center of the screen was a visual model of the vehicle, and to the right was a menu with different settings depending on where in the

navigation the user was. The concept also featured a "Library"-view that was opened from the right side. From this view, the user can open other vehicles previously configured. The concept also featured a merge-tool, used for copying values from a previously defined vehicle to the current one (see Figure 20).

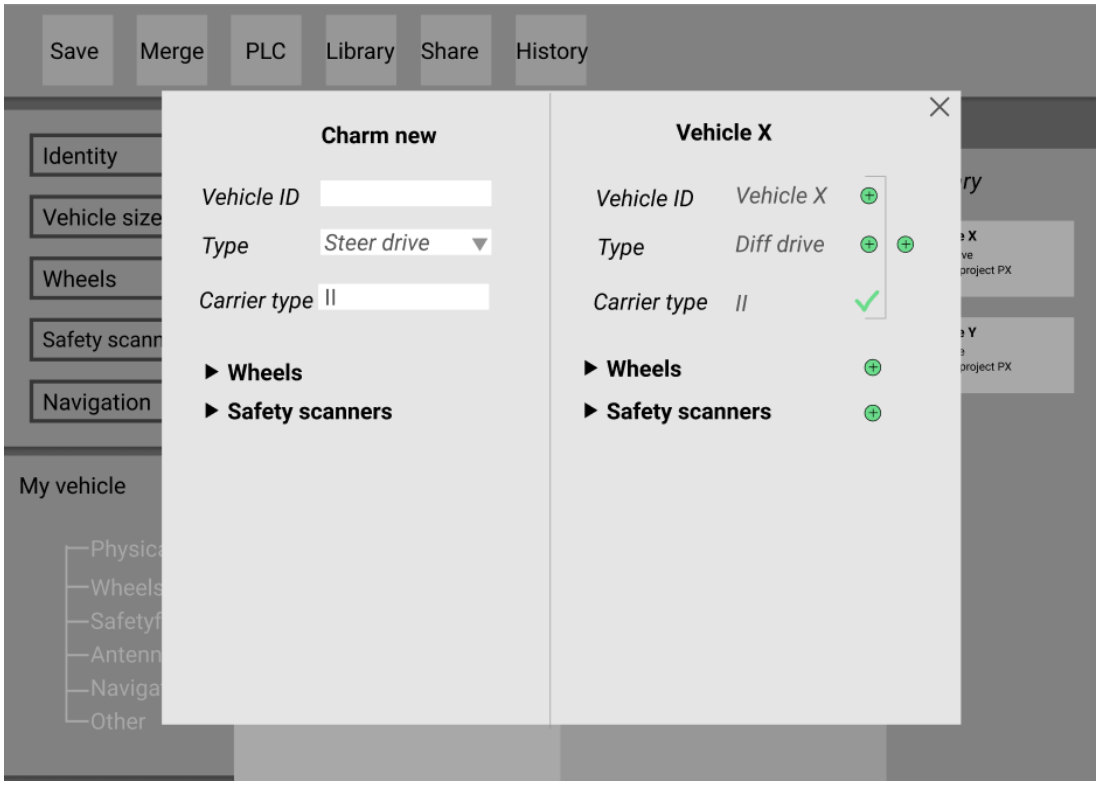

Figure 20: Prototype 1 merge view.

Evaluation
The concept was evaluated by interviewing one beginner user, one expert user, and one UX expert. The concept was shown and used as a basis of discussion to receive feedback.

The following feedback was received:

- The word "Library" does not convey what that view contains.
- It is not clear how to use the merge functionality.
- The design of the buttons panel might be unsuitable since it makes it difficult to expand the software if new functions need to be added.
- Some vehicles have motor parameters, this would be useful to add.
- It would be useful to be able to generate a list of components from the software.

7.2. Prototype 2

The second prototype was created based on Prototype 1, with changes made according to the feedback received, as well as UX and UI principles.

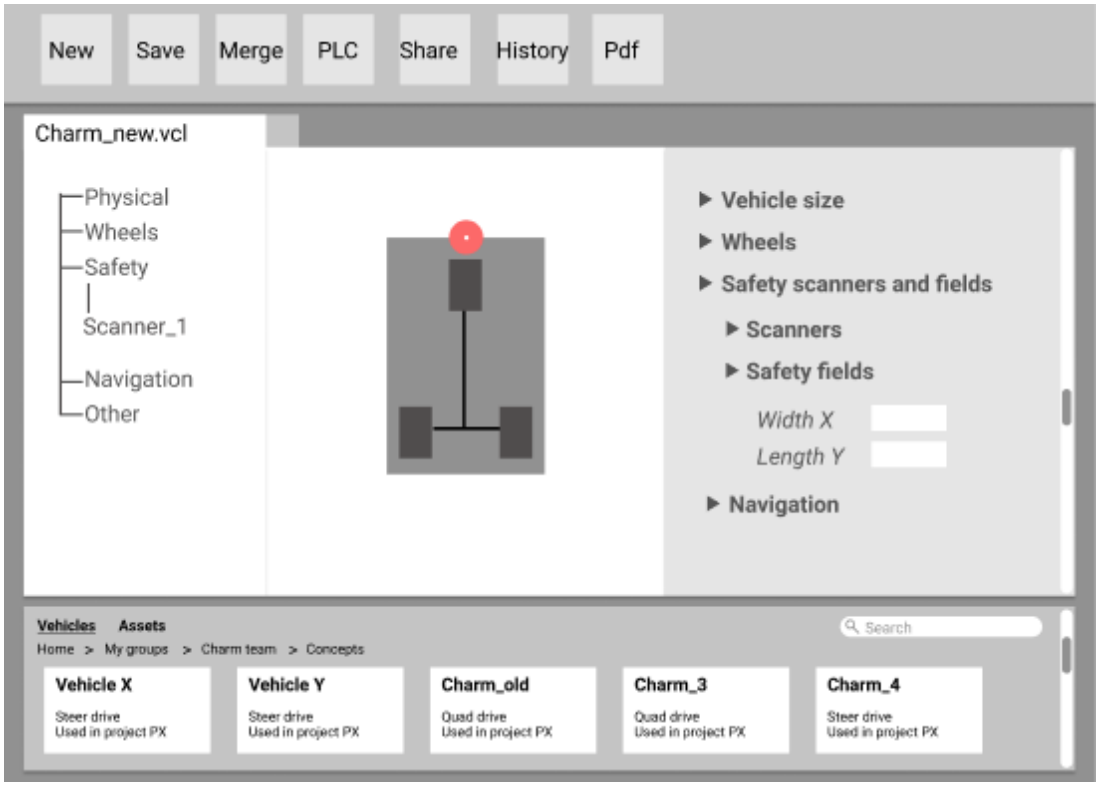

Figure 21: Prototype 2 safety scanners and fields view.

The left menu is a tree where all configurations that have been added are present, such as scanners and wheels. The middle view in a vehicle representation model. The right menu is an accordion menu with all settings present at the same time. This is both a means of navigation as well as input fields for the different parameters. The "Library" is renamed the "Vehicles/Assets" menu and is moved to the bottom of the screen, to utilize the screen better (see Figure 21). The merge menu is moved to the right side of the screen, with the mergeable vehicles available from the "Vehicles/Assets" menu (see Figure 22). Several vehicles can be merged into the current one simultaneously. The prototype also features a "History"-view, where the user can see changes that have previously been made (see Figure 23).

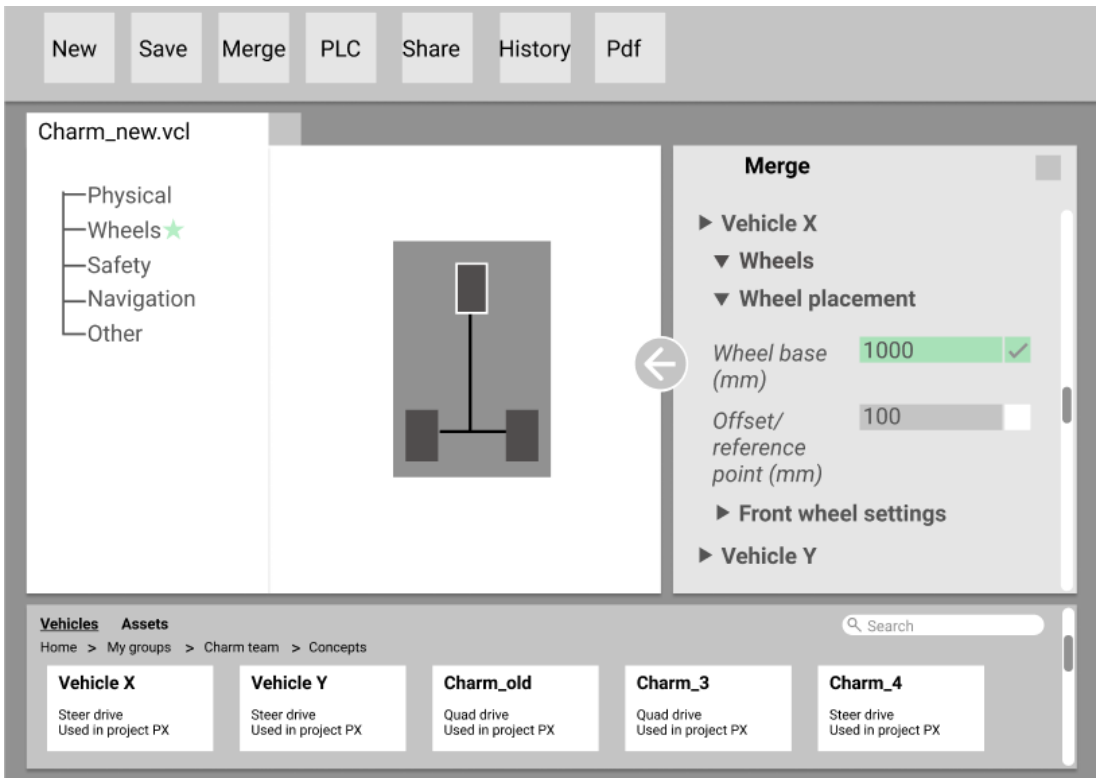

Figure 22: Prototype 2 merge view.

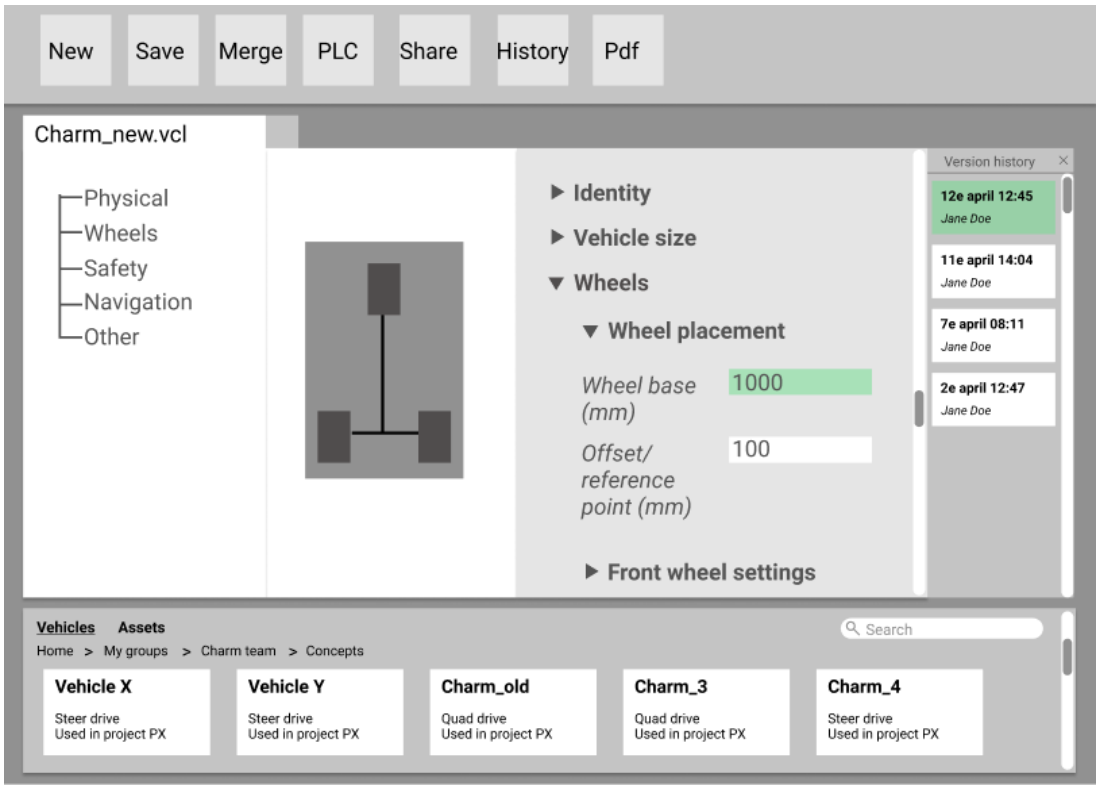

Figure 23: Prototype 2 version history view.

Test
An increment usability test was prepared in order to receive feedback from both beginner users (3 in total) and an expert user (1 in total) on the design. The users were asked to complete some tasks within the Figma model, and reflect upon what was particularly difficult or easy. The test included the following:

1. Navigating the studio view
2. Creating a new vehicle
3. Navigating the vehicle settings menu
4. Setting the size of the vehicle
5. Opening vehicles and assets from the lower bar menu
6. Adding safety scanners and safety fields
7. Viewing and understanding version history
8. Using the merge-function

Most of the tasks were completed quickly by all users. Two of the areas, 5 and 8, were difficult for most of the users. Task 5, the task of using the bottom menu bar for finding assets and old vehicles, was difficult to find for the users. Some misunderstood the button "Assets" for a subtext to "Vehicles". Most users looked first in the top menu to find Assets. Task 8 of using the merge function also caused some confusion for the users. It was unclear to them that the empty checkboxes were just that and not some other button or symbol. The word itself, merge, was also not clear for all users (of which none were native English speakers).

7.3. Prototype 3

After having tested the second prototype, a third prototype was constructed according to insights from the tests and some changes of concepts that needed testing.
The main insight from the test was that the "library function" was difficult to find, even though it was constantly located at the lower part of the screen, as depicted in Figure 21. The test subjects, in general, wanted to locate it via the top menu bar and expected this function to be more "hidden" than what it in fact was. It was also reasoned that the "library function" took up much space, and could benefit from being able to hide. Hence this was changed to Prototype 3 and the "library function" would be accessible both via the top menu bar and via a clickable tab icon at the bottom of the page (see Figure 24). The function was also renamed "Project explorer" since it was thought to be a more accurate description of the function rather than "Library". Because of the confusion regarding the hyperlinks "Vehicles" and "Assets", these were removed (see Figure 25).

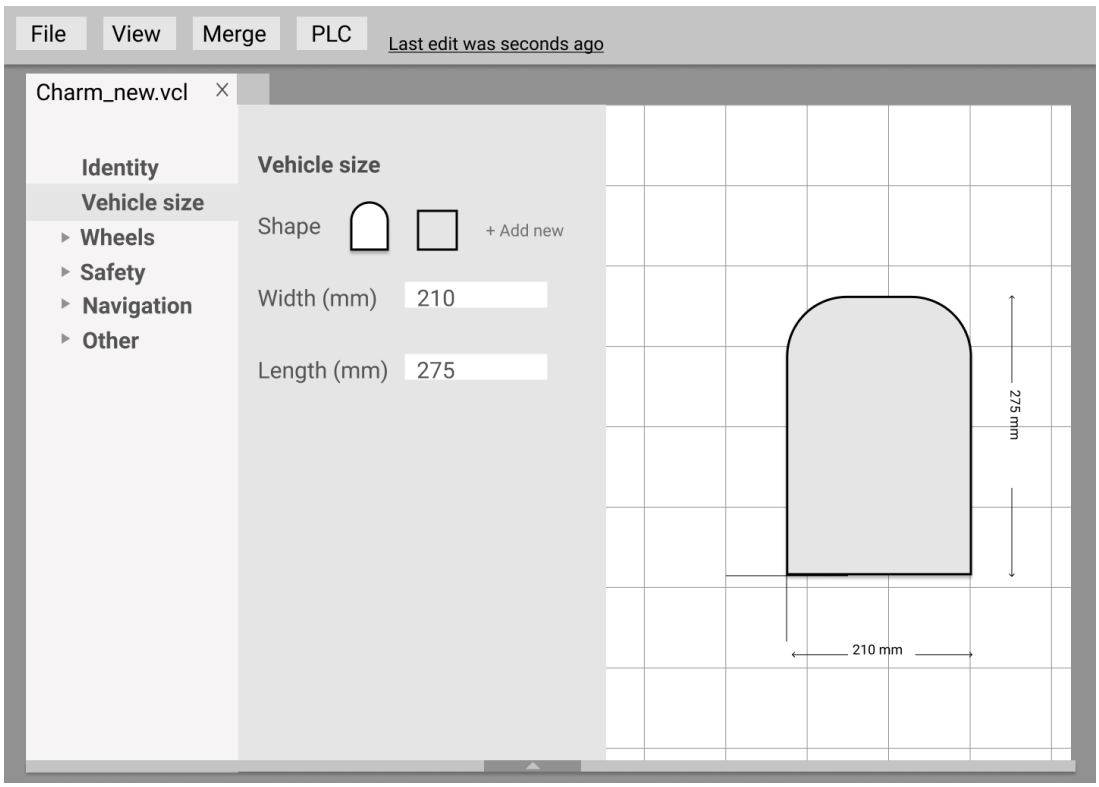

Figure 24: Prototype 3 size and shape settings view.

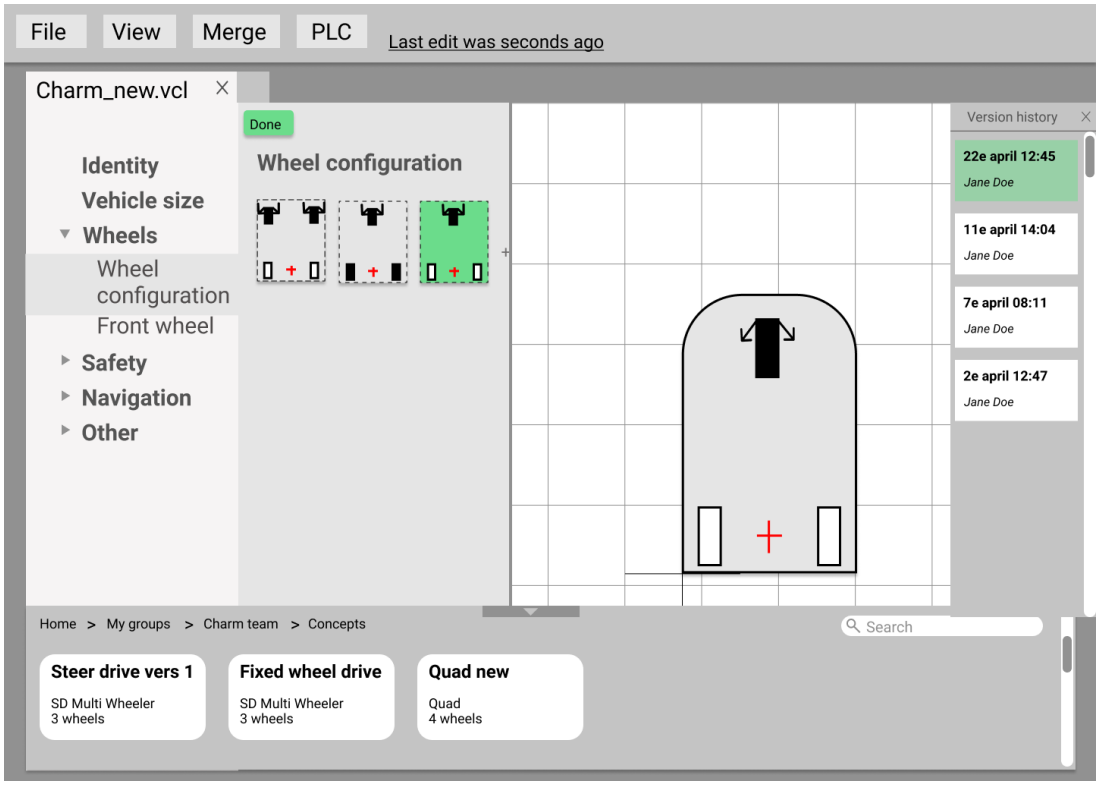

Figure 25: Prototype 3 version history view.

Another insight from the test was that it could be difficult for a user to not be able to see the shape of the vehicle that they were creating before adding both length and width in the software. In Prototype 2, if the user started by adding the width, this would have been represented by a horizontal line, something that some test subjects found difficult to interpret. This was solved by letting the user choose the "shape" of the vehicle first, before adding width and length (see Figure 24). This would allow the user to have a representation of the body of the vehicle right from the start.

Further, it was realized that Prototype 2 was not very scalable with the accordion menu as the main navigation element. With a large number of settings, this choice of navigation method would require a lot of scrolling for the user and perhaps not be effective. Hence, the structure of the interface was changed entirely by removing the tree structure and introducing a new way to navigate. The navigation method was designed with an accordion menu only for the setting headlines, such as "Vehicle size", "Wheels" etc, whereas the settings themselves in a separate menu to the right. The placements, right next to the main menu, were chosen to indicate that the two parts belonged together. The two menus were connected by a color indication showing in what part of the menu that the user was at all times (see Figure 25).

A change was also made to the "Version history". It was decided that the software should feature autosaving to prevent loss of work for the user. Inspired by Google docs, a hyperlink text was added to the top menu, saying when the last edit was made (see Figure 25). To cater to users expecting a "Save"-button, the choice to "Save to version history" was added to the top menu, under File.

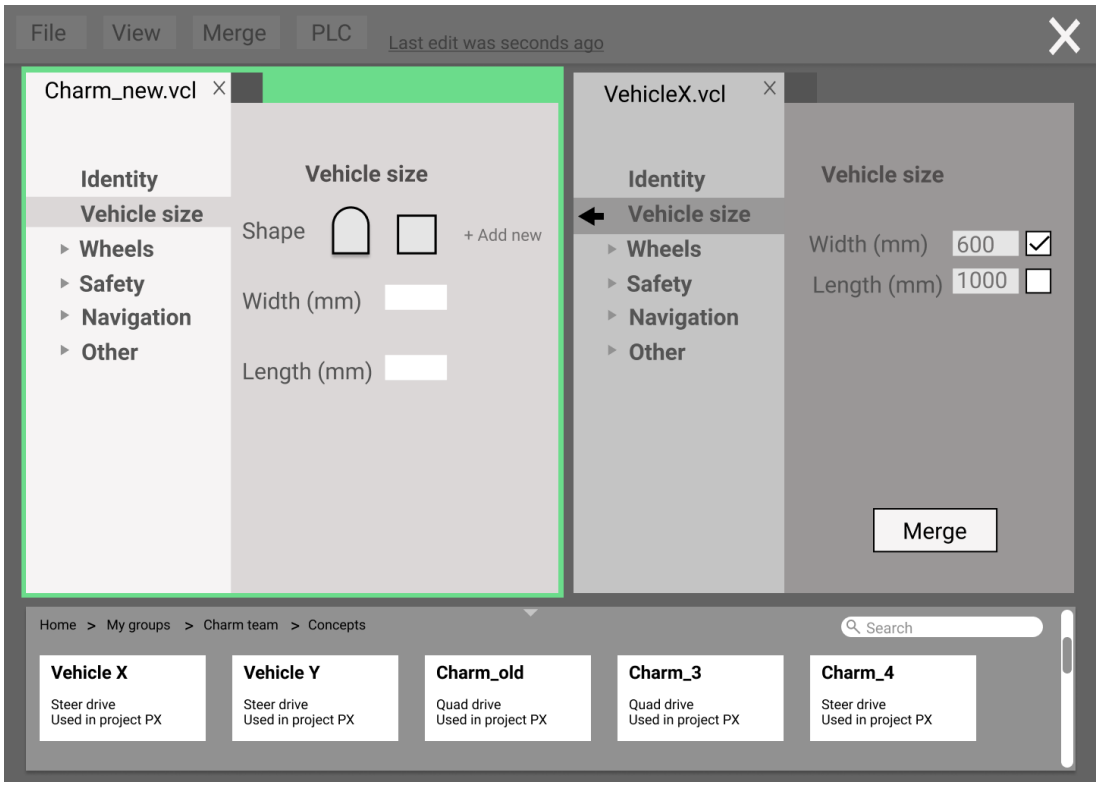

Figure 26: Prototype 3 merge view.

Lastly, the merge function was redesigned to better understand what was part of the original vehicle, and what was the merge window. The button was also redesigned to better indicate that it was a CTA. The checkboxes were also made more clear with a stroke added around them. Inspiration was taken from merge functions in some programming software that often features this kind of function and hence a black arrow icon was added to indicate what was being merged, resembling how the interface looks like in programming software such as Android Studio (see Figure 26).

Test
To test the redesigned Prototype 3, a test was constructed. The test was conducted with five new test subjects, all beginner users. The subjects were asked to complete some tasks in the Figma model, much like the previous tests. The tasks included:

1. Navigating the studio view
2. Creating a new vehicle
3. Navigating the vehicle settings menu
4. Setting shape and size of the vehicle
5. Finding and opening a vehicle from the project explorer function, and testing whether the name project explorer was descriptive.
6. Finding the version history view
7. Finding and testing the new merge function

Although most of the tasks were completed without problems. Some areas were discovered to be difficult for the users. When creating a new vehicle, some users had difficulties understanding the "add new" option when choosing wheelbase. And most users had difficulties finding the merge tool and project explorer without getting a thorough description of what these functions were. One user expressed that this could have to do with him not expecting these functions to exist at all and that this caused him not to look for a specific function to perform the merge or project explorer tasks, but rather trying to solve the tasks in another way.

The words merge and project explorer was still difficult for the users to understand, but the functions in themselves worked mostly well. The users understood how to use them, one user expressed that the merge function reminded him of the merge functions in programming software he had used and that this was a positive thing.

However, an issue with the merge function was discovered: The CTA button with the merge function was placed at the wrong location since the button needs to be visible even if the user emerges from multiple parts of the menu at the same time.

7.4. Prototype 4

The most dramatic change made to Prototype 4 from Prototype 3 was that color was added to Prototype 4. The colors selected were added from a specific pallet for Kollmorgen and were

not decided in this project. Some purely aesthetic details, such as rounded corners and drop shadows were also added to give the prototype a more finished look.

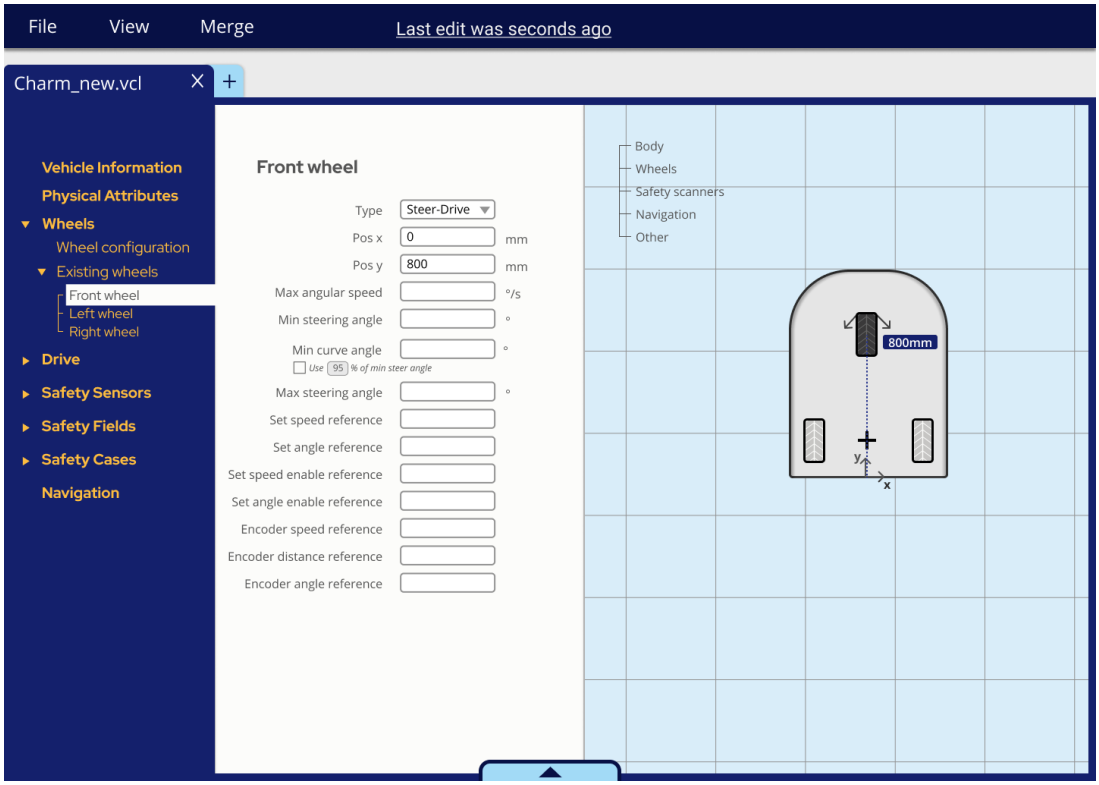

Figure 27: Prototype 4 front wheel settings view.

The overall structure with the headline menu to the furthest left and the settings in the area next to it was kept. However, to let the users navigate quickly to elements that they had configured in the program, the elements also showed up in the menu. For example, if a user had chosen a certain wheel configuration, the wheels would show up in the menu under a sub-category called "existing wheels" (see Figure 27). It was thought to be a convenient way to structure the configured parts for the user, but there were also some concerns about whether the user would find it confusing that the menu would work as both a menu and somewhat as the tree structure formerly had done.

To make the project explorer more prominent, the CTA button in closed mode was made bigger (see Figure 27). And the "assets" function, that previously let the user drag and drop certain elements such as wheel configurations into an active drawing was completely excluded. It was reasoned that the merge function would allow the user to do this if the user wanted. The project explorer remained hidden when opening the program, and was able to be found in the top menu under "view".

Instead of the option "Add new" for adding a new type of custom configuration, the function was called "create custom" (see Figure 28). This better signaled what it was and would not be mistaken for an "add" button.

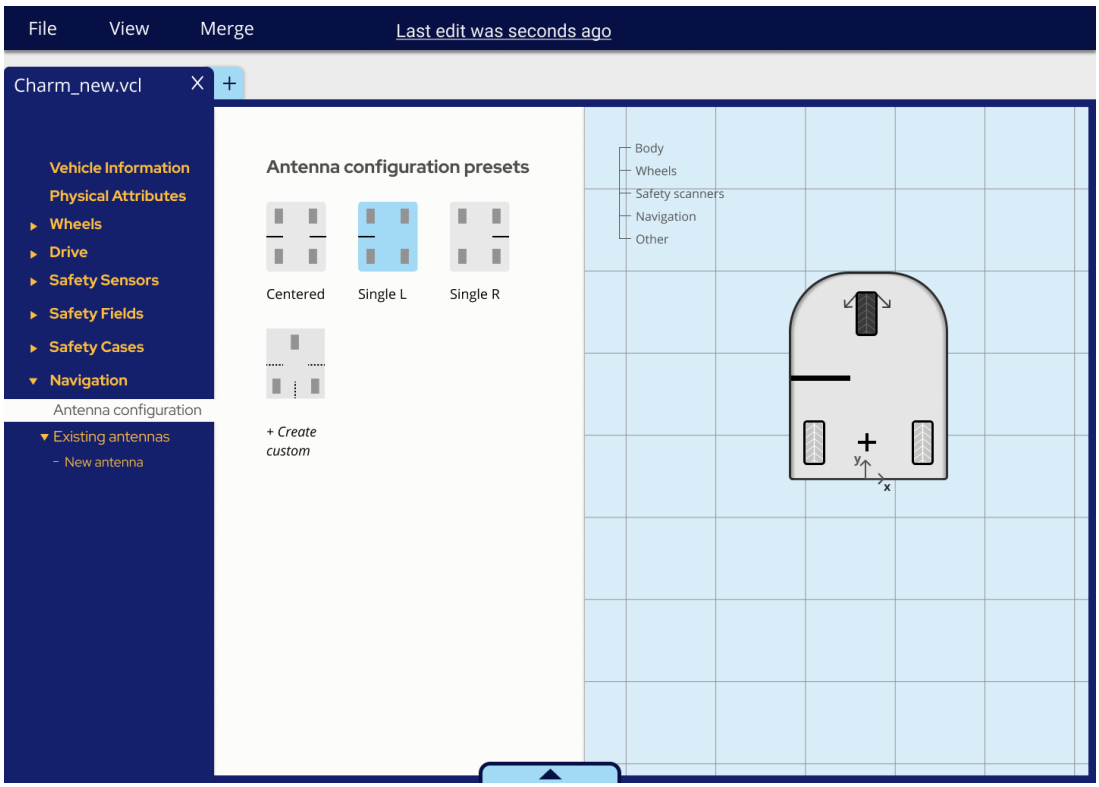

Figure 28: Prototype 4 antenna configuration view.

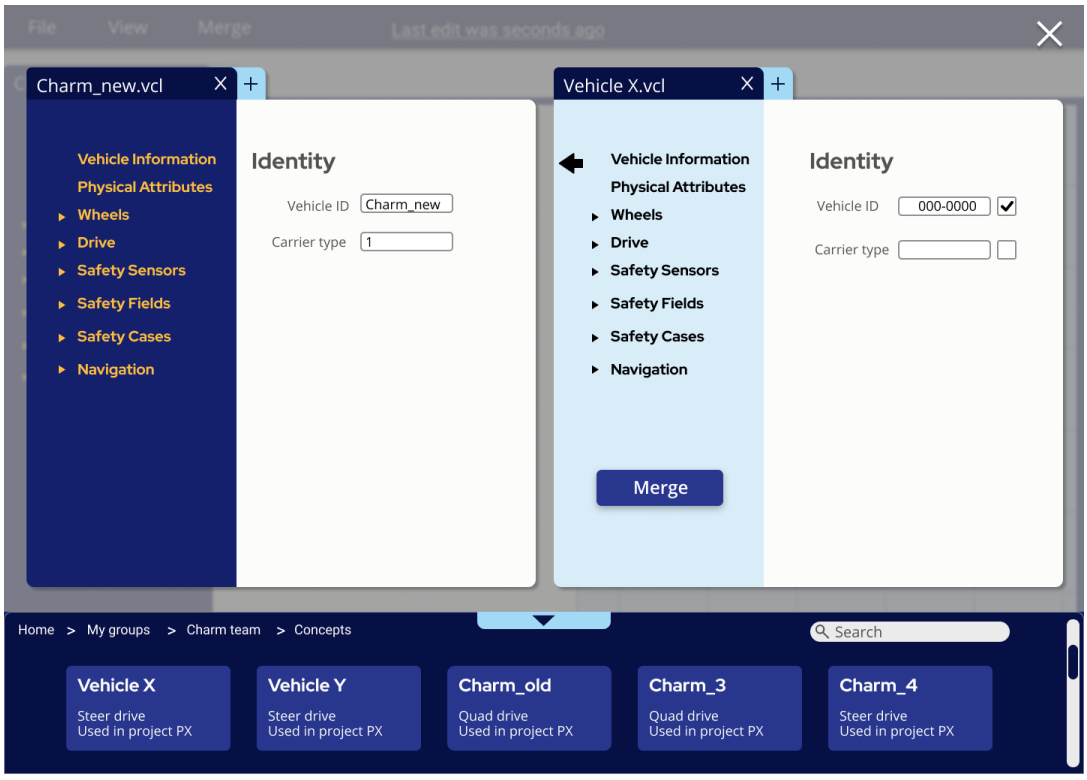

Figure 29: Prototype 4 merge view.

The CTA button in the merge view was moved to the menu since it better showed that the button would merge everything marked rather than the marked configurations of that specific page (see Figure 29). Merge and project explorer were still called that for Prototype 4.

Test

The tests of Prototype 4 were similar to previously conducted tests. The test was conducted with five people in total, where two were Kollmorgen employees and the remaining three people with no connection to Kollmorgen, but with a technical background. All the test subjects could be considered beginner users. The test consisted of several tasks that the users were asked to perform and included:

1. Navigating the studio view and finding the vehicle configuration program.
2. Creating a new vehicle, starting with vehicle information and physical attributes
3. Navigating the vehicle settings menu, including going back to configure multiple elements and vehicles
4. Setting shape and size of the vehicle as well as configure scanners, safety fields, and safety cases
5. Finding and opening a vehicle from the project explorer function, and closing the function
6. Finding the version history view
7. Finding and testing the merge function

The users were after the completion of the test of the prototype asked to discuss what emotions they experienced throughout the usage using the Geneva emotion wheel as a means for discussion and rating the intensity of the emotion.

The test subjects in general were able to complete the tasks with little struggle. They quickly learned how the program functioned. The feared issue with confusing the multiple-use properties of the right menu seemed to be no problems for any of the test subjects which was a relief. Most of the test subjects were prone to not use the illustrative representation as a means for navigation within the program, which further strengthened our belief that it was good to include to be able to navigate to already configured components via the menu. The illustrative representation was used mainly for receiving feedback about the inputs given to the software.

It was made clear that the history view should include in-text what was changed, and not only by marking the changed parameter. None of the test subjects were able to point out what had been changed when asked a question about the version history function. It was also made clear that the user needs a clearer type of feedback when actively pressing save in some of the configurations, and not only display this by adding the saved element in the menu.

Most of the test subjects found the prototype harder to use in the beginning of the test than in the end of the test. A more clear explanation when first opening the program should be included to redeem this. But partly could also be explained that the users got more used to the prototype as the test proceeded. The tasks were in many ways repetitive since the

program works similarly throughout the usage.

The test subjects in general did not expect the function of autosave to be included.

Success rate
To evaluate the improvement from the existing KMA software to our prototype, a success rate test was performed. At the beginning of the project, the test was constructed and tested with beginner users for the existing Kollmorgen software. When Prototype 4 was finished, the same test was performed for this. Some tweaking of the questions had to be done, but in essence, they tested similar things and were of similar complexity.

The test was done by defining 17 different tasks for the users to complete in the respective software. The requirements for partial success and total success were defined beforehand (see Appendix A). The performance was scored with 1 point for total success, 0,5 points for partial success, and 0 for failure. The mean score for each task was calculated by taking the mean average of all the participants' scores.

In 12 of the 17 tasks, the mean score was higher in Prototype 4 than Layout Designer (LD) (see Figure 30). The most notable improvements were made in the areas of visualizations (tasks 3 and 7), safety fields (tasks 13 and 14), and safety cases (tasks 15 to 17). In one of the tasks, task 1, the score was notably lower for Prototype 4 than for Layout Designer.

Task nr	Task description	Mean score (LD)	Mean score (prototype 4)
1	Create a steer drive vehicle	1	0,4
2	Create another steer drive vehicle	0,83	0,8
3	Interpret vehicle and wheels visualization	0,75	1
4	Create a diff drive vehicle	0,83	0,9
5	Open a vehicle from another project	0,25	0,5
6	Add an antenna	1	1
7	Interpret antenna visualization	0,1	1
8	Move the antenna	0,9	1
9	Delete the antenna	1	1
10	Navigate to safety sensors	0,92	1
11	Add a sensor	0,75	1
12	Move the sensor	1	0,9
13	Find where you can add a safety field	0,75	1
14	Change the size of the safety field	0,33	0,8
15	Change the safety case max velocity	0,92	1
16	Set the PLC bit 3 to "ON"	0,08	1
17	Activate the safety field	0,4	1
	Overall mean score	0,69	0,90

Figure 30: Success rate result

Emotional response

The prototype was also compared to the KMA software in terms of emotional response. To do this, a Geneva emotion wheel was utilized when asking participants to rate to what intensity they felt the different emotions.

Emotion	KMA software	Prototype 4
Interest	3,7	4,4
Amusement	1,8	3,2
Pride	2,3	2,4
Joy	2,0	3,2
Pleasure	2,0	1,8
Contentment	2,2	2,8
Love	0,5	0,6
Admiration	1,7	1,2
Relief	2,3	2,2
Compassion	1,3	1
Average positive emotion	**1,98**	**2,28**
Sadness	1,5	0,2
Guilt	1,0	0,2
Regret	0,3	0,6
Shame	0,8	0,8
Disappointment	2,0	0,2
Fear	2,2	1
Disgust	1,5	0,2
Contempt	1,5	0,2
Hate	1,0	0,2
Anger	2,3	0,2
Average negative emotion	**1,42**	**0,38**

Figure 31: Geneva emotion wheel result

As can be seen from Figure 31, the users reported experiencing lower levels of negatively associated emotions during the usability testing of Prototype 4, compared to the KMA software. They also experienced higher levels of positively related emotions. Some of the comments were:

- "It felt nice when it behaved like I wanted it to"
- "You feel content when you see the visuals"
- "I felt relieved when I understood the system"
- "I was a bit nervous since this is all new. But it was easier when it turned out to not be any problems."

Feedback from expert users (application engineers)

To get feedback from expert users before the final design was constructed, the same usability test as the one conducted with beginner users for Prototype 4, was conducted together with

three expert user (application engineers). However, the test was used more as means for discussion than testing the usability than for the beginner users.

The expert users had overall less trouble than the beginner users finding different functions in the program, likely because they where more in general familiar with the terminology used as well as the underlying theory and practice of AGV systems. Generally the solution was received well and much of the feedback regarded technical details, the major takeaways from these sessions were:

1. The image of the vehicle representation must be rotated so that the direction of travel was along the X-axis according to the current standards
2. Have all elements emanate from the reference point, and not from a separate coordinate system
3. Grouping the functions "Safety Sensors", "Safety fields" and "Safety Cases" under one headline "Safety" to decrease over cluttering the menu
4. Restructuring the Menu so that the headline "Navigation" was before the headline "Safety" since this better follows the workflow of today
5. Exclude the subheadline "existing", such as "existing wheels" and just have the added elements directly under the headline since the sub-headline was considered confusing
6. Splitting up some of the parameters under "Wheel" and place them under "Drive"
7. The program must support being able to import safety sensors and antennas
8. Include ability to write a short description of the vehicle under "vehicle information"
9. Short descriptions of different functionalities while hovering as well as some considerations about specific names of functions

The expert users also stressed the importance of designing the solution in such a way that it would be easy to add functionality later, since new functions are likely to be developed, or other functionalities moved into this part of the software system. Furthermore, the users also brought up some functions from the current software that was not included in the prototype, that they would like to see included in the final software.

Feedback from in-house UX experts

The evaluation of Prototype 4 also included two feedback sessions with in-house UX experts at KMA. These users also got to perform the usability test on Prototype 4, but as in the case of the expert users, the test acted more as means for discussion for the UX experts. The UX experts were less successful at completing the tasks than the expert users, which was expected since their knowledge about the software, terminology, and experience in using the software was lower. The feedback mainly consisted of views on the UI:

1. Exclude the element tree in the vehicle representation window since the menu already covered this functionality.

2. Redesigning the project explorer so that it opens up from the tab instead of always being present in the UI
3. Grouping the functions "Safety Sensors", "Safety fields" and "Safety Cases" under one headline "Safety"
4. Reconsider the word "existent"
5. Merge function could be replaced by copy /paste
6. More clear names on functions
7. Splitting up some of the parameters under "Wheel" and place them under "Drive"
8. Clearer indication of what element is being configured in the vehicle representation window
9. Left align all text
10. Chunk text in areas with much information, meaning to present information in pieces rather than in one large sequence

One of the UX experts also expressed an urge to save after changing a parameter in the configuration window, even after knowing that the program was thought to use autosave. This could show that the prototype had some issues regarding feedback on input parameters. It was also expressed that the "add" buttons that were used to save some of the settings, such as for "safety cases" and "safety fields".

8. Deliverables

The result of the project is a set of guidelines for what to consider when designing a vehicle configuration software, and a proposed design for a vehicle configuration software for the KMA system. Both are based on the user needs found through the studies of the project.

8.1. Design guidelines

From the user research made, the following guidelines were created. In addition to these, it was found that internal consistency and feedback were important needs to consider. However, since they are valid in general for all kinds of software, they will not be further discussed.

Include all static settings - The main problem of the Kollmorgen software is that there exist multiple vehicle configurations in different software. This is time-consuming, creates confusion, and is a potential source of error. All static settings, settings that do not change during run time, should be located in the same software. This also gives an overview of the vehicle and how different settings relate to each other.

Make the solution scalable and allow for the number of parameters to grow as the technology evolves - Functionality is constantly being added to an AGV system, and the software tools have to match that. The more difficult it is to add additional settings and new features to the software, the faster it will be unable to fulfill the user needs. The new software should be scalable so that it is easy to build upon it. For example, the menus should be organized in such a way that it is easy to add new items.

Avoid exports and imports - Each configuration file should be accessible for all software that needs it, without the need for importing a local copy of it. One of the problems with the Kollmorgen software is the excessive need for importing and exporting across software. This is time-consuming, frustrating for the user and increases the risk of errors.

Having only one file that other software and/or users can access by reference also makes the version handling better. This makes it possible to track changes.

Allow for copying settings between vehicles - When configuring a new vehicle, users rarely start from scratch. They usually base the new vehicle on an old one, and apply changes, or make a new one but copy parts from other vehicles. To support the need for using parts of different vehicles, it should be easy to copy both individual settings and groups of settings from one vehicle to the other.

Place settings in normal use order, but allow free navigation - To help beginner users, the settings should be placed in an order so that you can follow them step-by-step. This would reduce the need for a tutorial, something that can make the software seem more difficult to use than it actually is [6].

However, it should be possible to navigate freely between the settings. Since there are many different ways the users interact with the software, there should be flexibility in the usage. For the advanced user, it should be possible to skip a setting or go back and change it later.

Limit the number of visible settings - With complex software with many settings, there is a risk that the user will experience cognitive overload. To reduce this risk, the design should be minimalist and not contain irrelevant information [1]. For example, settings that are normally not needed can be hidden under "advanced settings". These could be deprecated settings that only apply to older vehicles, or settings that can be automatically set but in some cases need to be customized.

Use visualizations for feedback - Visualizations help the user understand what parameter is what, especially when it comes to physical dimensions. Visualizations should be updated when an input is made to better match reality and to offer feedback when a change has been made.

Explain difficult concepts - When it comes to complex, highly technical systems like this one, there may be many parameters that are difficult to understand. The primary goal should be to use names that are self-explanatory. These can be accompanied by visualizations. In those cases where it is impossible to sufficiently explain by just the names or visualizations, an explanatory text should be offered to avoid confusion. A problem with the Kollmorgen software is that some variables are unknown even to expert users, and some can have different meaning depending on who you ask.

8.2. Final design proposal

The final design proposal is a desktop software concept named **Vehicle Configuration**. The design is based on Prototype 4, described in section 7.4.

Overall layout

The layout of the main view is structured in three parts: a left-side menu for navigation, a middle menu with all parameters, and a visualization of the vehicle (see Figure 33). The left-side menu is structured in the same order that users would normally access the settings, in line with the usability heuristic of **Match between system and the real world**, described in section 3.4. However, the user is free to navigate the software as they choose and is not forced into a certain workflow, following the guideline **Place settings in normal use order, but allow free navigation** described in the previous section. The accordion menu allows for more headers being included when the need arises, in line with the guideline **Make the solution scalable and allow for the number of parameters to grow as the technology evolves**.

When the user makes a change to the parameters, the vehicle representation is updated accordingly. Having this form of visual feedback for a user, and that in a simplistic way represents the vehicle and its components, satisfies one of the major needs that users expressed: getting feedback that input parameters are correct and that the final vehicle ends up as the user had imagined. This is in line with the usability heuristic **Visibility of system status**, as well as the guideline **Use visualizations for feedback**.

Project explorer

When first navigating to the start screen, the user is met with the **Project Explorer** (see Figure 32). Here the user is given a choice to open a previous project or create a new one. The two choices are differentiated by colour and placement and a plus sign. There is also a

search bar, where the user can search for projects and vehicles. In this way, the user is greeted with a choice and not with an empty start screen. The user is also able to open new tabs directly, where the user is met by the same interface as in the beginning for consistency. This structure was inspired by browsers such as Firefox and Google Chrome, where suggestions for recently opened websites are shown as cards on a dashboard. This makes the design consistent with other software the user might use, in line with the usability heuristic **Consistency and standards**.

By introducing this type of working, the user is able to quickly locate old projects, and also have multiple vehicles open in tabs at the same time. Since users often prefer using old projects for guidance and using multiple templates or vehicles, it was important to create a structured way for the user to keep track of old projects, and let the user interact with multiple vehicles at the same time.

Physical attributes

When further navigating in the software, to **Physical attributes**, the user can make multiple choices (see Figure 33). The first parameter is "physical attribute type". This is a new function that is thought to replace "carrier type" of the current KMA software. During the Empathize phase, it was discovered that there existed a need for being able to categorize vehicles based on physical attributes other than those that are configured. This is why this function was kept but with modification. "Carrier type" has been used in two different ways. First, it has been used to define different behaviours at a site, and in this case, the carrier type of a vehicle can be changed during run time. Secondly, it has been used to categorize vehicles based on physical attributes such as height, which can affect the routes the vehicle can take at a site. It is this second function "Physical attribute type" is thought to replace, in line with the guideline **Include all static settings**. The function of dividing vehicles into behavioural characteristics is recommended to exist within another software dealing with the dynamic characteristics of the vehicle and the site, such as Layout Designer.

The shape of the vehicle is set in this view, as is the size. The reference point is also set in this view. It is a point on the vehicle, usually in the middle of a wheel pair, from which all other measurements emanate.

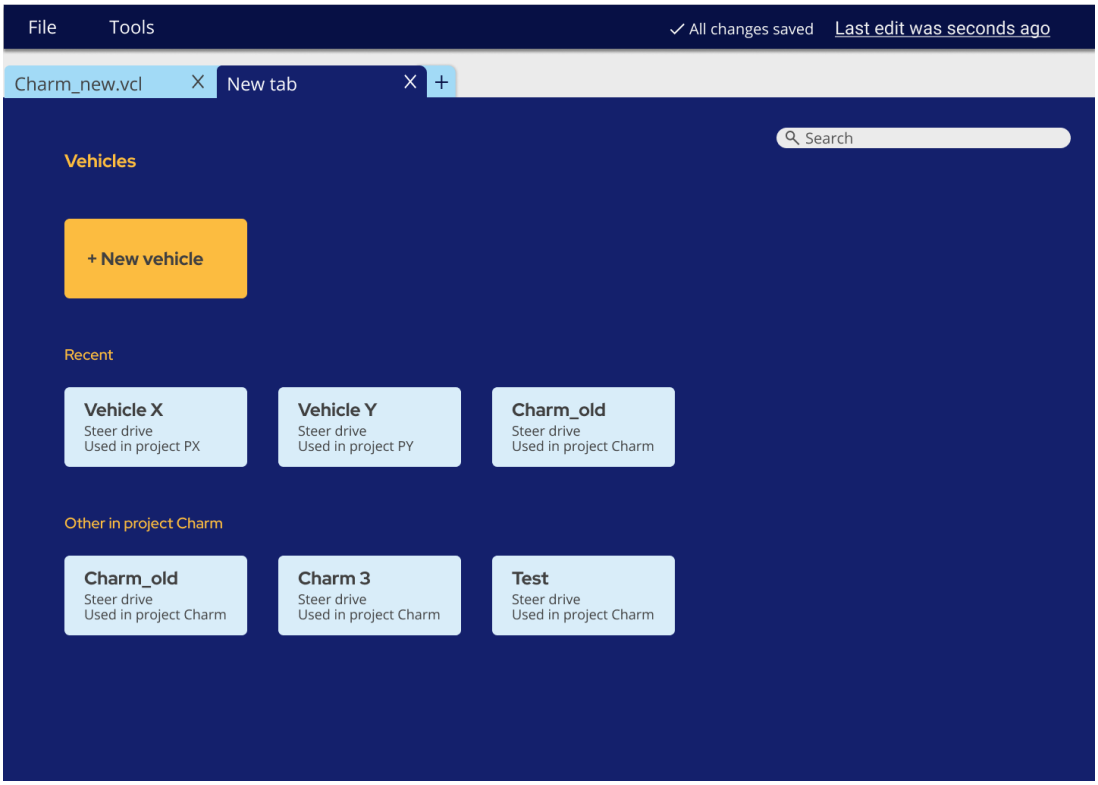

Figure 32: The Project Explorer

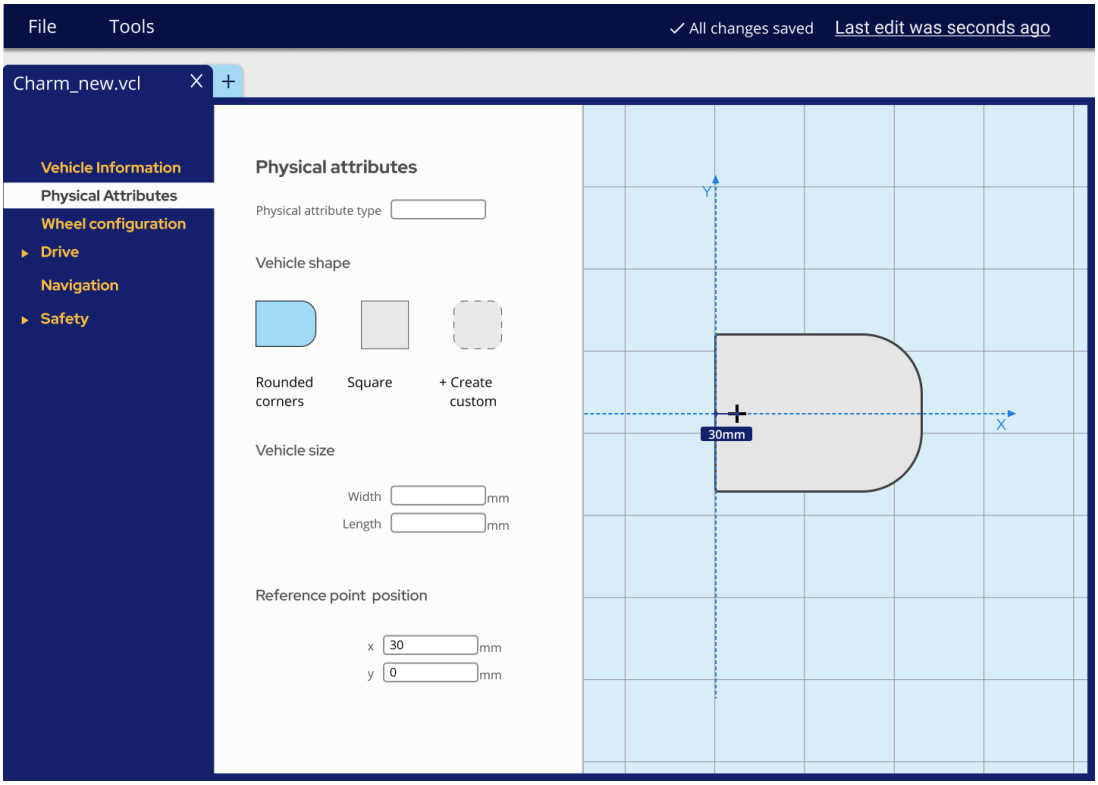

Figure 33: Physical attributes view

Wheel configuration

The user sets up the wheel configuration by choosing one of the presets or creating their own configuration (see Figure 34). The user receives feedback of their choice by the chosen preset being marked in blue, and that the wheels appear on the vehicle representation. This type of feedback was also designed with the user need of receiving feedback on input parameters and selected choices.

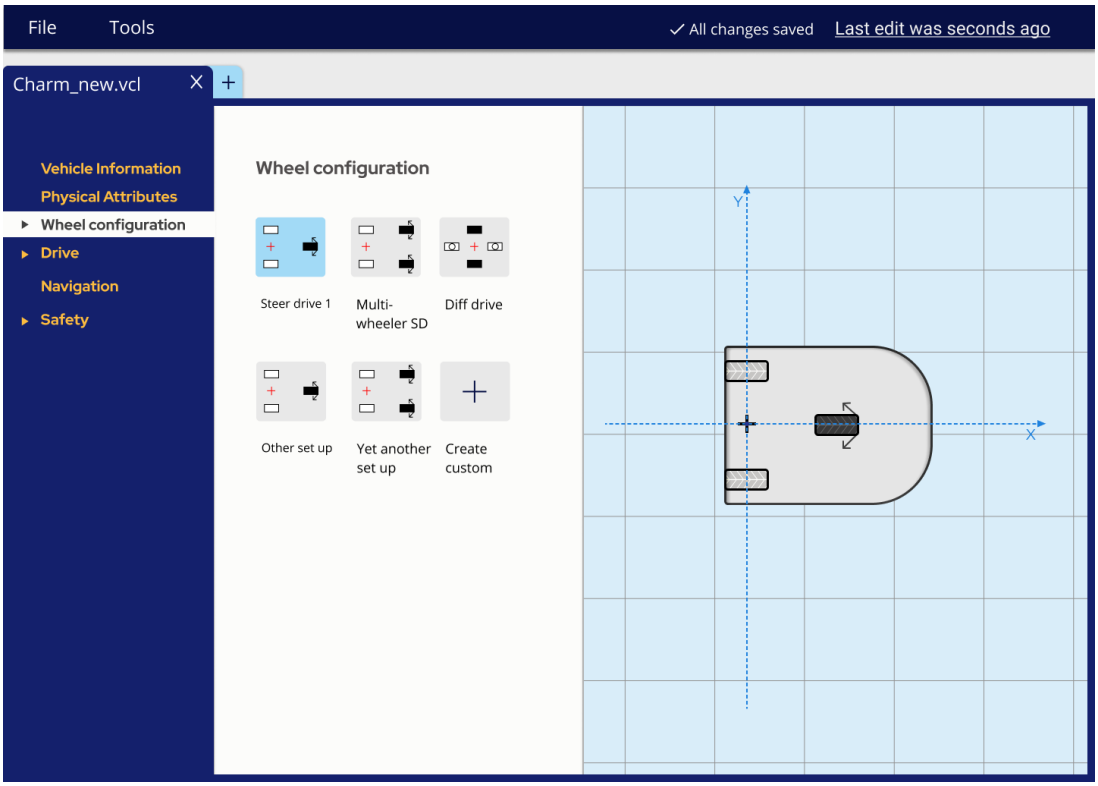

Figure 34: Wheel configuration view

When the user has chosen a configuration, the associated wheels can be found in the menu to the left (see Figure 35). When clicking on the headlines in the menu, or on the wheels in the image, the options for that wheel are shown. The wheel that the user is changing is also marked in blue, so that the user can quickly get an overview of the vehicle, and what part the parameters belong to.

Furthermore, the functionality of moving one wheel at the same time as another wheel (interlocking the wheel with another wheel) was also implemented under the wheel options (see Figure 36). The parameters that are related to the drives of the wheels are found under the headline "Drive". There each drive is connected to a certain wheel.

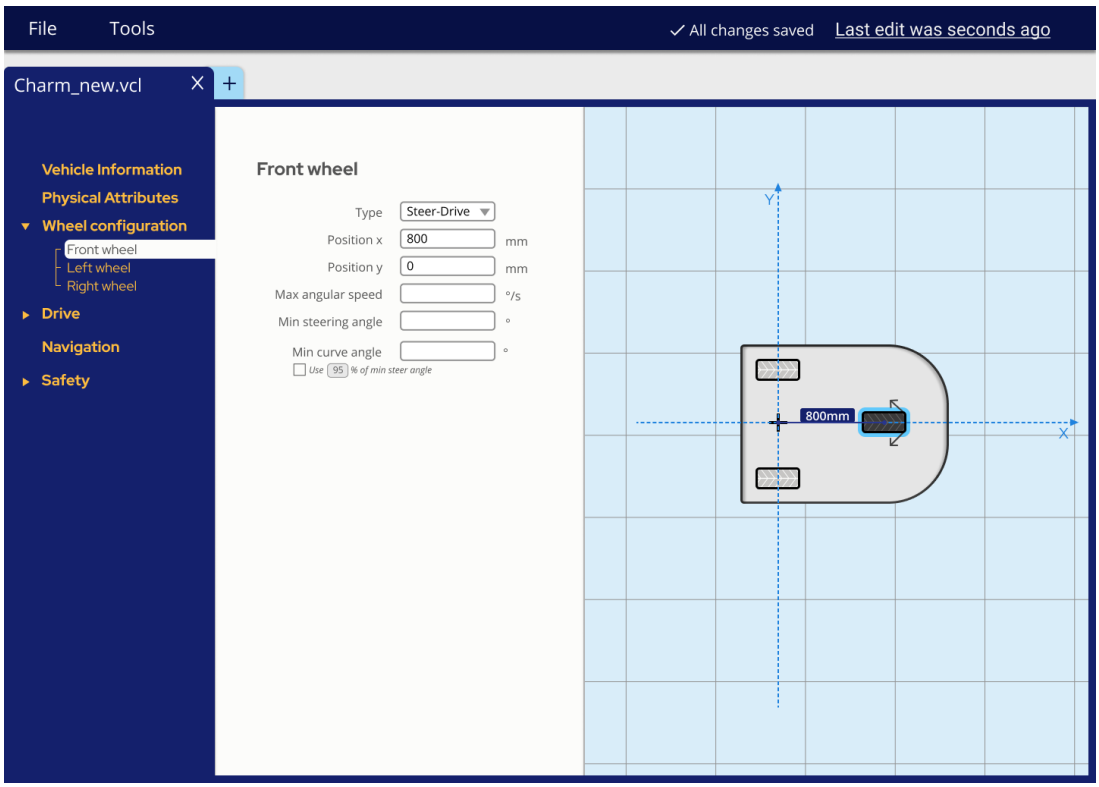

Figure 35: Settings for the front wheel

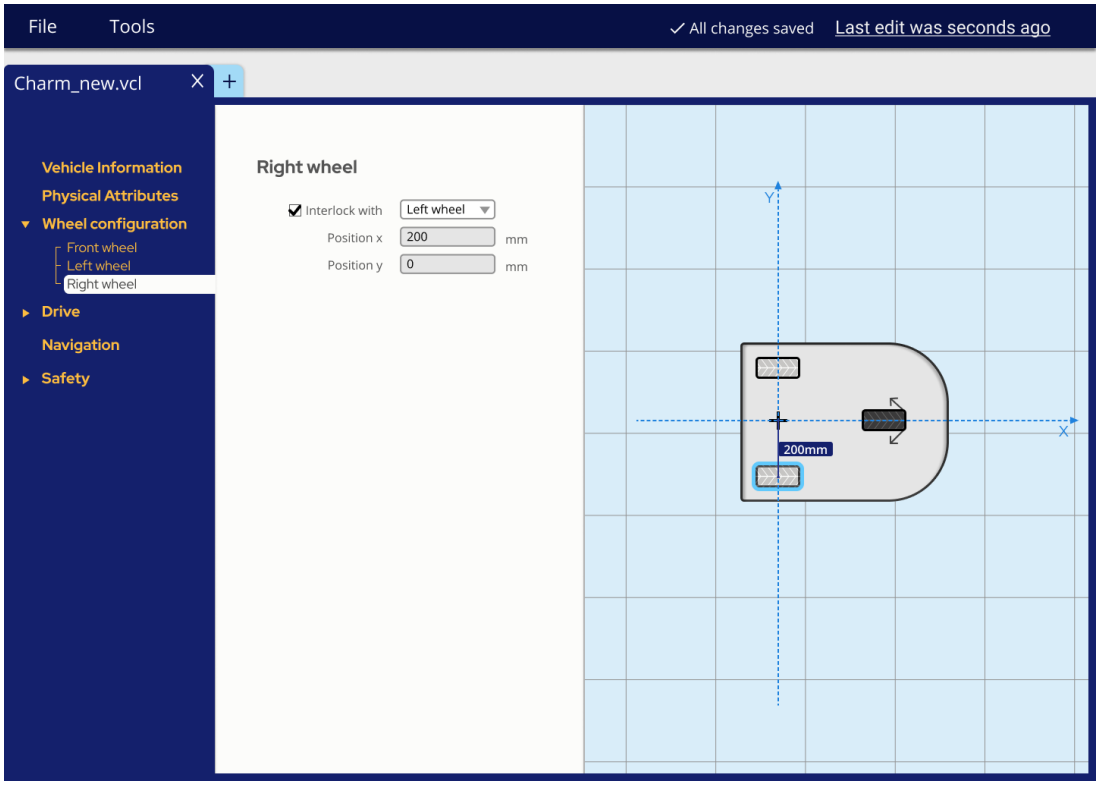

Figure 36: Settings for the right wheel

Navigation

When choosing a navigation method, the user gets to choose from different options by interacting with checkboxes (see Figure 37). The checkboxes are used in favour of, for example, radio buttons since the navigation can be a combination of different technologies, and radio buttons would signal that the user can only choose one of the methods. In other parts of the software, radio buttons were more suitable and were implemented. By choosing suitable CTA representations, the software becomes more clear and easier to understand.

When the user has chosen a navigation method, the options for that navigation method appear together with the option to "Add new" under "Navigation" in the left side menu. In the example in Figure 38, only the options for configuring an antenna is shown and not other options for other navigation methods (see Figure 38). This is a way to only show relevant information to the user, in line with the guideline **Limit the number of visible settings**. Furthermore, if the user clicks on "Advanced settings", the physical length and width of the antenna can be configured. However, these are parameters that are rarely changed since they are set automatically depending on which model is chosen. Therefore these settings are not shown to the user by default. By only presenting the information that is useful to the user at given moments, it reduces the information that is presented to the user and increases the clarity of the software.

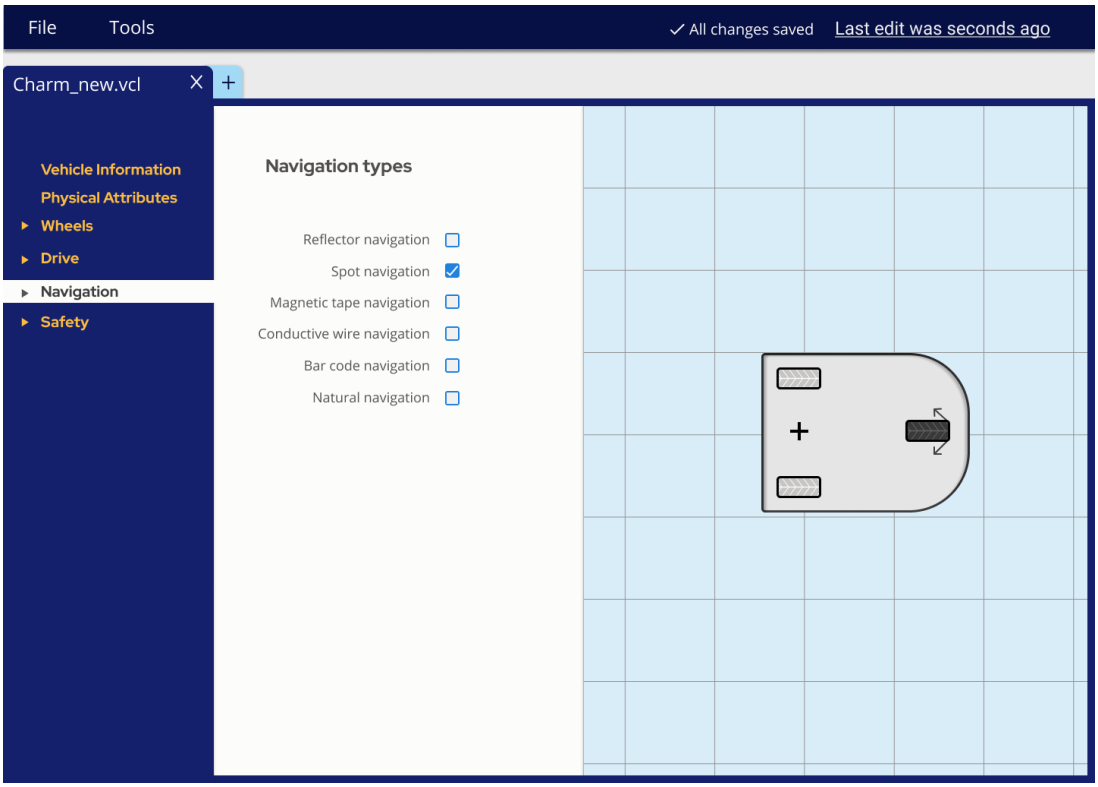

Figure 37: Navigation view.

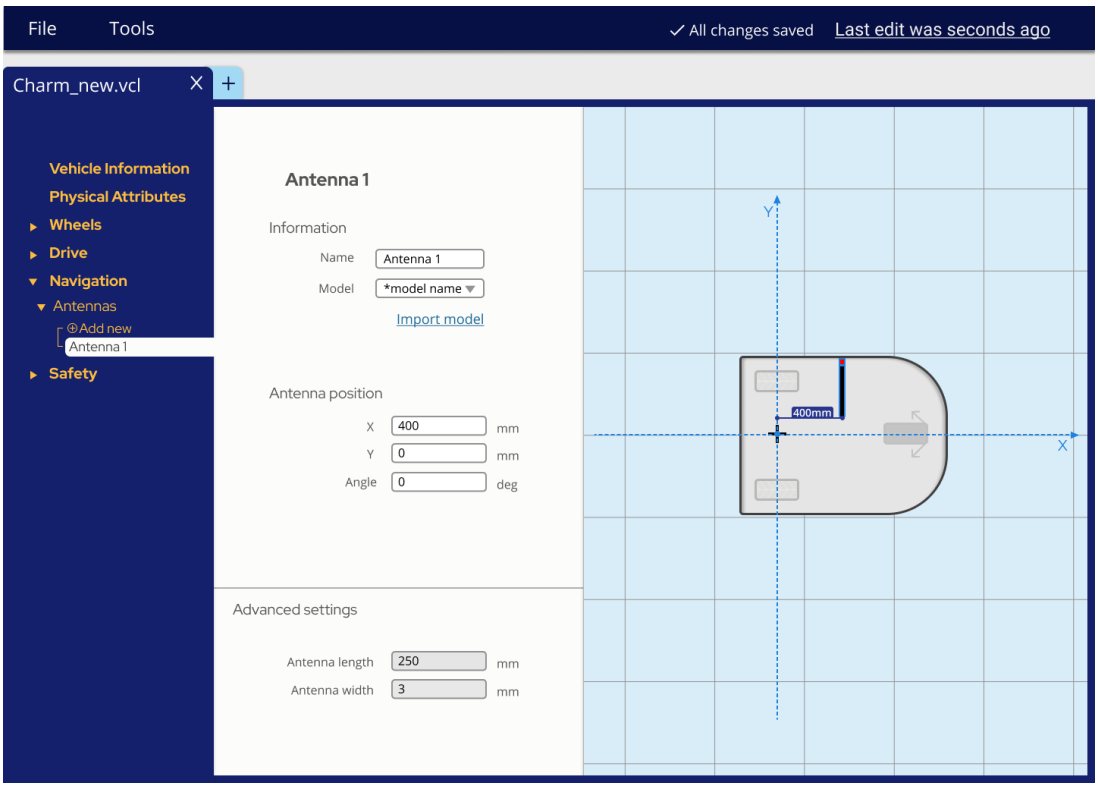

Figure 38: Antenna configuration view.

Safety

The safety system consists of safety sensors, safety fields and safety cases. Sensors are configured in the same way as antennas with model and position (see Figure 38). The sensors have safety fields associated with them; these are configured under safety fields. They are added in the same way as antennas and sensors, and then accessible from the left menu. The safety fields are associated with a certain sensor, which acts as its origin (see Figure 39). The size of the field is set, as is the offset relative to the sensor. The safety cases are also added in the same way. They are configured by what velocity and angle they are valid for, what PLC bits are active, and what safety fields the case will use (see Figure 40). The PLC bits are set to active or inactive, as well as on or off (1 or 0). Toggle buttons together with radio buttons are used for this purpose. The user can either click the toggle button first to activate the bit, or click the "on" or "off" button directly. This will activate the bit, and at the same time, set whether it is on or off. It is also set what safety fields of what sensors should be active for that case. Several fields can be added. When one field is chosen and added, it appears in a list. It then becomes possible to add another one. The active fields are visualized in the vehicle representation in a similar fashion as when configuring them (see Figure 39).

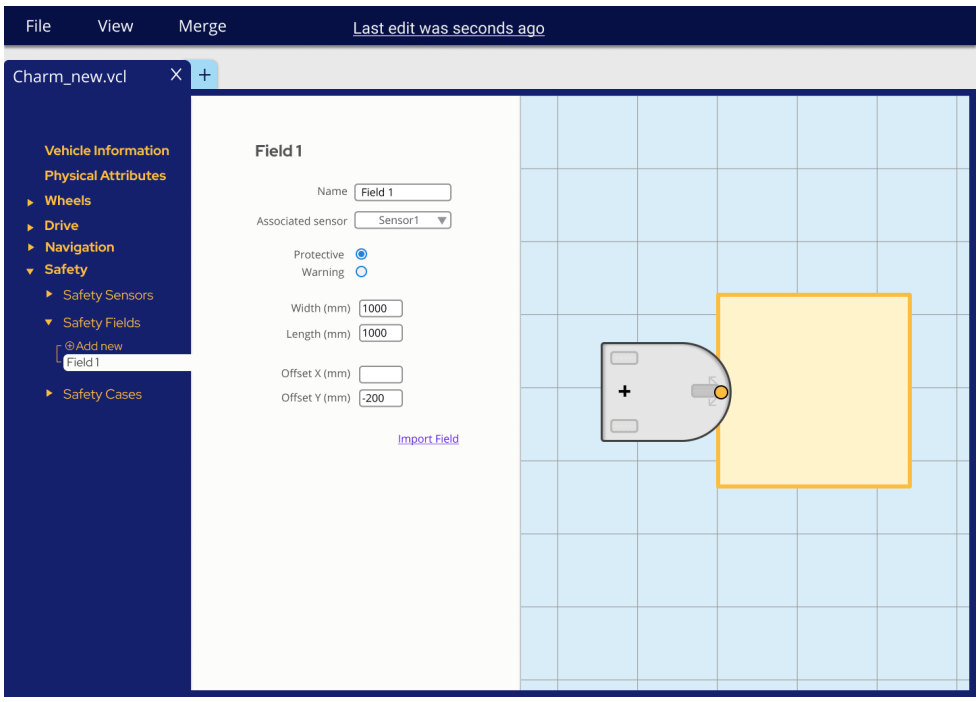

Figure 39: Safety field configuration view.

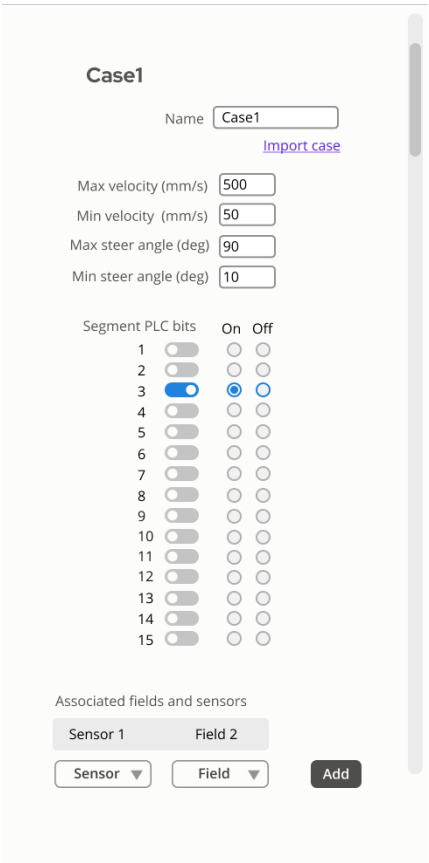

Figure 40: Safety case configuration view.

Merge

A user insight that was found early on was that most users start with an old vehicle configuration and make changes to it, or copy settings from it, when making a new configuration. This is what led to the guideline **Allow for copying settings between vehicles**. This is what the merge function does. The user navigates to it via the top menu; under "Tools", there is a "Merge" function. They then choose what vehicle they want to copy parts from by a project explorer. They then check the boxes of everything they want to copy. They can check each parameter separately or check the headlines to include everything under that headline. To merge the selected parameters into the vehicle configuration, they click the Merge-button (see Figure 41).

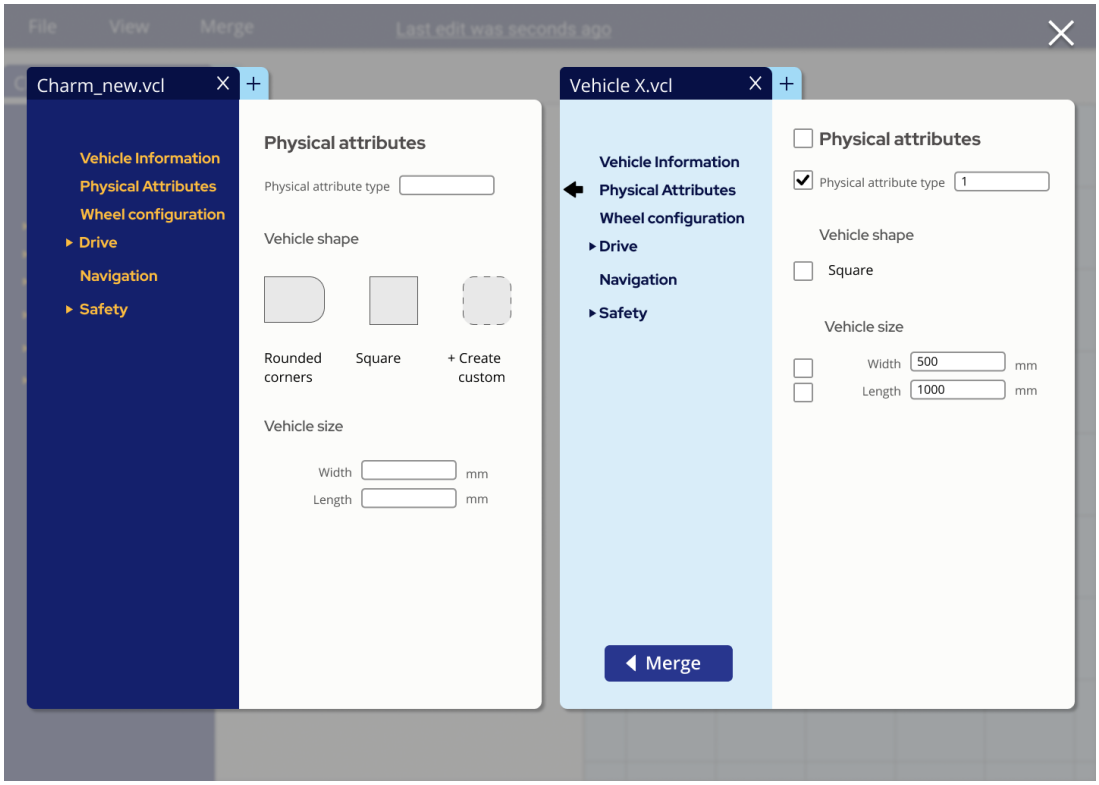

Figure 41: Merge view.

Version history

Using a version history functionality enables users to track changes, which is helpful when troubleshooting an issue. It also allows them to see who made a certain change, so they know who to ask for advice on that topic. It is also helpful to be able to go back to an earlier version if a mistake has been made in the current one. The version history function includes dates, the possibility to name different version, and shows the user who made that change (see Figure 42). To enable version history, the software is recommended to exist within a larger software universe, where other software and users access the configuration file by reference. This would also enable the guideline **Avoid exports and imports**.

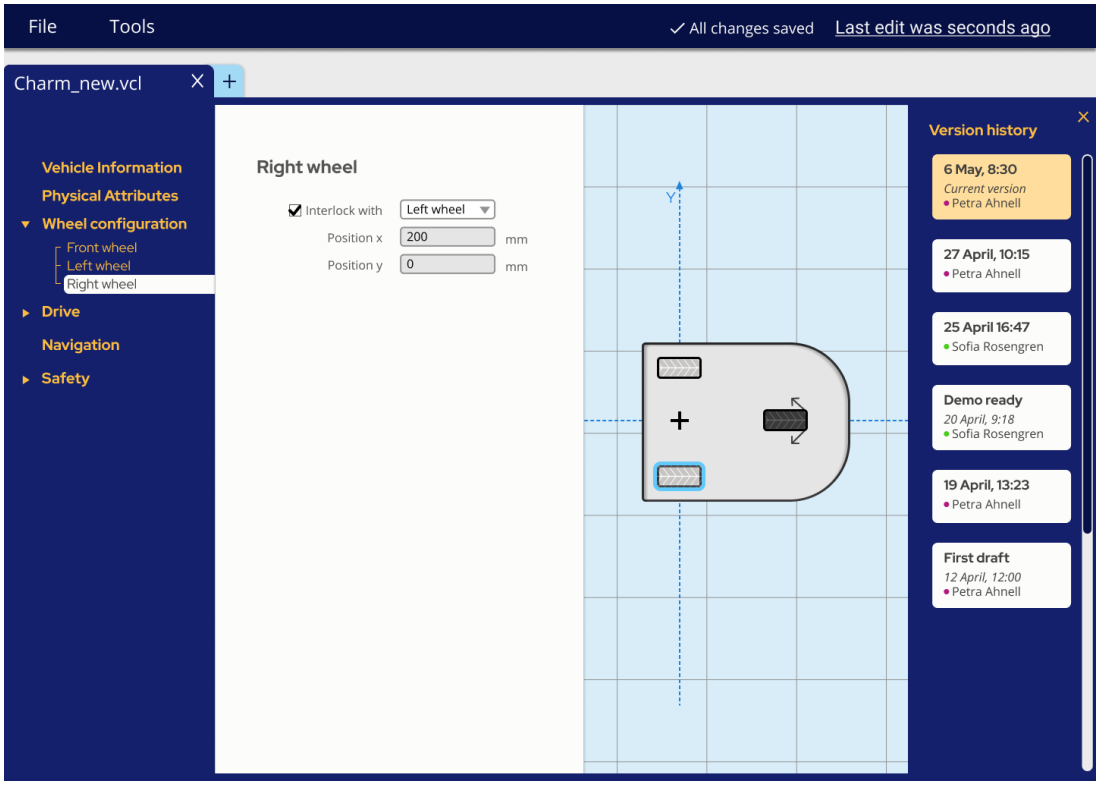

Figure 42: Version history view.

Changes made from Prototype 4

The final design was based on Prototype 4. Most of the functionality was kept, as well as the layout and colouring, but some changes were made. One of the biggest changes was that of the Project Explorer. Since the previous solution to opening a project explorer at the bottom of the screen had not been intuitive for most users, the concept was reworked so that the project explorer is opened when opening the software or a new tab.

Another thing that changes when compared to Prototype 4 is where the reference point is set. After feedback from expert users in Prototype 4, it was made clear that it is convenient for expert users to keep all measurements emanating from the reference point. In Prototype 4, most measurements emanated from a coordinate system with the origin in the center back of the vehicle. Since the reference point hence gets a more central role, it was crucial that it was set earlier in the workflow.

Furthermore, the vehicle has also been rotated so that the direction of travel is along the X-axis. This choice derives directly from feedback from expert users on Prototype 4, where the users said that it was crucial to maintain the orientation so that the direction of travel was along the X-axis due to the fact that users are accustomed to this and that the other KMA software relies on the vehicle being oriented this way.

A new menu item, "Drive", has been added. Since some wheels have multiple drives, and some have none, it was reasoned it made sense to make it separate. As a result of this, the number of parameters in the settings screen for an individual wheel has been reduced since many of them have been moved to the corresponding drive settings screen.

The merge function (see Figure 41) was kept mainly the same as before since it had gotten good feedback from both experts and beginners. The checkboxes were moved from the right side of the options since it was thought to be more clear. A "check all" box next to the headline was also introduced, as well as an arrow icon on the "merge button" for the ability to merge whole "parts" of a vehicle.

9. Discussion

Generally, the process and methods used were successful and worked well for this project. The combination of using beginner users for usability tests, but reconnecting with expert users allowed using the expert users' knowledge in the most efficient way possible, but still unveiling some of the more obvious usability flaws of the old software. Some parts of the process and methods, result and future work, as well as a social sustainability aspect is worth discussing more and are mentioned in the following chapter.

9.1. Process and methods

9.1.1. Transcription as a part of qualitative studies

Early in the project, it was decided that all interviews with expert users should be transcribed mainly to be able to use citations from the transcriptions in the KJ analysis during the "define phase". Since it was thought that much of the design decisions were going to be made on the qualitative data found in the interviews with the expert users, the goal was to document the data as well as possible. Even though transcribing was time-consuming, many hours were invested into the transcriptions, in hindsight, it can be concluded that the time was well invested.

The transcripts are likely to have kept the research rather unbiased and held the researchers back from interpreting the answers during the interview while taking notes. It kept the research accurate and true to what the users said. Transcribing (as well as the video recordings of the expert users using the software) also helped us understand the more complex technical details of the software and usage, and it was helpful to have the transcription to find information. During the actual act of transcription, it was also possible to find some details that neither one of us had picked up.

9.1.2. Geneva emotion wheel

The decision of using the Geneva emotion wheel (GEW) was based on the wish to broader the perspective from the usage of the software than mere usability. Using an emotional evaluation method at the end of a usability test did surprise some of the users, and some users were initially skeptical about the method.

Something we noticed was that the way the participants interpreted the scale was very different. Some of the participants expressed the value 4 as "rather high" and others considered this value to be a "low value". Hence, the quantitative results of the method might not be accurate as in being able to tell the reader that a certain software evoked a certain feeling to a certain degree. However, it does provide the reader with some more general insights such as "The software generally evoked more positive than negative emotions".

However, since the participants knew that they were being tested on a concept that the researchers had developed, and were asked to evaluate the emotional response by the researchers, it might have skewed the results in favor of the final design concept. The participant might have, even though being asked to judge fairly based on their personal experience, given higher ratings on the final concept than they felt. This either to please the researchers or that they knew that the software was new and they found it interesting to test something they knew few others had tested.

The main benefit of using the GEW after a usability test was that it was an effective way to get users to reflect upon the usage. Users would talk about specific situations during the test and would say that they did especially dislike or like that part, and the model did provide some useful qualitative data that could be used in a KJ analysis. It was also an important insight that a user could have performed well according to the success criteria set up for the success rate metrics, but still reported feeling insecure when asked to discuss the usage of the software afterward. For future usage of this method, we would therefore recommend researchers to focus even more on the discussion around the emotions when using GEW and to set aside time for more discussion than on the rating of the emotions.

9.1.3. Remote user studies

Due to the ongoing Covid-19 pandemic, all the user studies were performed remotely. Both interviews and user tests were performed via Microsoft teams or Zoom and the participants were asked to share their screens. The remote user studies did not impose any issues. We believe that especially for working professionals, it is a lot easier to accept an invitation to a remote video meeting than it is to find time for a physical interview. And users were also able to accept meeting invitations on shorter notice when asked to review parts of the concept for instance.

For the tests performed with the beginner users, the fact that the tests were performed remotely might have also in fact had a direct positive result. The participants were able to sit in the comfort of their own homes and might have not felt as "observed" as in a physical test where the facilitator sits in the same room. It could have benefited users during the tests. However, while performing usability tests remotely clearly was no obstacle in this project (it might even have been beneficial) it is also important to remember that the reason why it was so easy to perform the tests remotely was that the project revolved around a graphical interface. Had the product developed been a physical object, it might have caused more trouble than benefit to performing remote usability testing.

9.1.4. Product development mindset

In classic product development, there is an emphasis on gathering information at the beginning of the project, before any real prototyping starts. In software development, it is more common to start making basic prototypes very early in the project. When making a digital product, it is easy and cheap to make prototypes early and often. The benefits of prototyping early are the possibility to get user feedback faster, to faster understand what works and what does not work. Possibly, the design could have been refined further if we had started prototyping activities earlier, having time for more iterations of user feedback. However, the project was not only about designing a new and improved interface, but also about understanding the underlying needs of the user and constructing guidelines. These results might not have been as good if not the first part of the project had had a strong emphasis on understanding the user.

9.1.5. Early concepts

It can be noted that the final design proposal has a general structure similar to the first two designs that were made, **Matlab** and **Tree structure**, (compare Figure 33 to Figure 13 and

Figure 14). These were conceptualized based on the ideas generated by the Brainwriting session, and rough designs were made using Figma. Having made these designs so early in the Ideate phase may have affected our view on what the design should look like. After having made them we had more brainstorming sessions and combined different ideas into concepts using a morphological matrix. These were then sketched on paper. To be able to make a fair comparison, it might have been better to just have sketched the first two so that all could be compared on an equal level. Making only a rough representation of each concept makes it easier to not get too attached to a certain concept too early on in the process.

9.2. Result

9.2.1. Success rate

The success rate was used as a metric to see how the proposed design was measured compared to the current KMA software. The result showed that the new design proposal was better. However, with small test groups such as these, the numbers should not be given too much weight. The result may also be affected by the fact that the questions had to be adjusted slightly since the two concepts were so different. Also, the layout of the concepts in relation to how the questions were posed might affect the result. For example, in task 1 the design proposal scored significantly lower than the KMA software. This could be explained by the fact that all settings required for the task were in the same screen for Layout Designer, and that this screen was already open at the beginning of the task. In Prototype 4, the required settings were in several different views. The structure of the prototype with these different views is crucial for the easy-to-navigate interface, and thought to overall increase the usability of the interface, only not for this specific task.

According to the Nielsen Norman group, the results from this study can also be regarded as significantly higher than the average value [8]. This is thought to depend on two things. The first is that the participants are all either technically skilled or have knowledge about KMA. This can cause the user to perform better than the "average" user met with these types of tests. However, more importantly, the tasks were thought to be rather easy for the users. Therefore, we do not believe that you should compare the results of the success rate from this study to results from other studies, but that the results can rather be used to compare round one and two in this particular test.

9.2.2. Generalizability

This study has only looked at the vehicle configuration made in KMA software, and the needs of the users using KMA products. Therefore it is not certain that the design guidelines that have been constructed will apply to vehicle configuration software of other AGV systems. However, another system of similar complexity and with similar users is likely to face the same challenges. For example, some issues are believed to have sprung from the fact that the system has grown organically and has not had a major remake while functions have been added. This could be true for many systems that have grown over time. Settings such as wheels, navigation method, and safety system are something all AGVs have, even though there might be differences to what configurations can be handled by the system. With the guideline "Scalable solution", the design becomes more flexible, and differences in AGV systems can be handled in the same way as adding or changing features of one system.

9.3. Future work

This section will describe some remaining questions to be answered for the future, and suggestions for future work.

- Since only a selection of parameters was implemented in the design concept and tested in the usability test, there is still a need to test the functionality of the software concept when more parameters have been implemented. The software is built to be expandable, and new parameters should be able to be included easily. However, it would be interesting to see how the users perceive the software as a whole when more parameters are added. Would the result be the same or would the users perceive the whole concept as more complex?

- Some functions are not fully developed in the concept, and hence have not been tested. Among the functions that need to be developed for the software to function properly, creating a custom wheel configuration or creating a custom shape are worth mentioning. The idea of being able to choose presets of wheel configurations and preset shapes of the vehicle also indirectly includes being able to create an own configuration or shape if the user would not find a suitable preset. The functionality of creating these custom "presets" has not been tested or developed.

  Another function that has not been implemented, is to be able to export a components list as a PDF. This functionality is in the final concept placed under file and export as PDF, but the PDF in itself has not been designed or structured. Both these functions could be something to look further into and develop further in a future project. Furthermore, auto-save is a functionality that has not been tested. The function did not get much response from the expert users, and it would be interesting to hear if any expert users would be directly opposed to implementing auto-save.

- Changes from Prototype 4 to the final concept have not been tested in usability tests. It has received feedback from UX experts and employees at KMA but still requires testing to assure the usability is good for beginner users. It was reasoned that the changes from Prototype 4 to the final concept were not very drastic and that due to lack of time it was not prioritized to do a final test. This is something that is needed in the future.

9.4. Social sustainability and automation

Social sustainability is sustainability in terms of humans and regards issues such as gender equality, human rights, and future generations and aims to ensure a good quality of life for all humans and the generations to come [7]. The term as well aims to manage business impacts on people. Companies have a direct or indirect effect on social sustainability as they affect employees, are drivers of development and indirectly affect every person and customer in the value chain [9]. It is, therefore, important to discuss what effects on social sustainability this project may have.

AGV technology does substitute manual forklift drivers in warehouses, and our project may be directly connected to that process. The vehicle configuration could ultimately be connected to the replacement of manual forklifts and drivers, and hence harm social sustainability.

However, there are plenty of reasons to encourage increased automation in the industry. Increased automation leads to people not having to do repetitive, unergonomic, or unhealthy work tasks, and promises of people having time to do less work, and have more time for leisure. In our specific case of easing the process of replacing manual forklifts with AGVs, it provides a more ergonomic environment for the people working at the warehouse, as well as a dramatically reduced number of lift accidents due to use errors of manual lifts.

We believe that automation has and will continue to provide positive outcomes for humanity as a whole and decrease the number of people who work in unsafe or unhealthy environments and that our solution can be a part of that positive development. The promises of every automated innovation, however, we believe rely much on how the positive outcomes are distributed among people. The solution to this should not be decreasing the development of automation, since the ultimate benefits are plenty and contribute to a socially sustainable future. Making sure that people who suffer from layoffs due to automation get sufficient support is one way to mend this issue. Another mentioned by United Nations Global is by creating decent jobs [9].

Usability is a way to mend the gap between the highly technical and advanced automation industry. By enabling user-friendly tools, that ultimately would need less and less technical competence is hence one way to ensure that people can continue to work in the development of the AGV industry, without necessarily having to have obtained a degree in a technical subject. The product and theory presented in this project might be a small step on the way, but it is a step nevertheless.

10. Conclusion

The aim of this project was to investigate the user needs and form a user journey for engineers working with vehicle configuration in the AGV industry. The research questions were the following:

- What does the user journey when setting up a new AGV system look like for application engineers today?
- What are the needs and requirements of the users of today?
- How can the vehicle configuration software be improved in terms of usability?
- What is important to consider when designing a vehicle configuration software?

The user journey is more closely presented in section 5.5. The usage of the vehicle configuration software is likely to happen at the start of the user journey, when creating a new vehicle. However, the user is likely to use the software multiple times later during the user journey when updates need to be done. This is especially true when talking about users who work for companies providing tailored AGVs to their clients.

The user needs regarding what the software lacks in usability and usage are presented below:

- The most important information should be presented most visible
- Images and visualizations must be clear and serve a purpose to the given situation in which they are presented
- The user should receive feedback when making an input
- Allow sharing and reuse vehicle configurations
- Allow for communication between KMA, customer and end-user The solution should behave in a consistent way throughout the usage
- Ease usage for beginner users
- Allow for iterative work for easier copy/ paste options and allow for users to update multiple configurations at the same time

Regarding pure functionality and functions that should be included, the needs were:

- Allow the user to divide vehicles into different groups
- The wheel configuration and wheel positions can be set
- Means of navigation, such as antennas or reflector scanners, can be configured in the same software but also be imported
- The sensor locations can be configured
- Safety fields can be configured, but also imported

The expected results were to formulate guidelines that could be used for designing a vehicle configuration software for the AGV industry, as well as a design concept prototype of such software. The final concept is presented in chapter 8.2. A more in-depth description of the design guidelines can be found in section 8.1. The guidelines are presented below:

- Include all static settings
- Make the solution scalable and allow for the number of parameters to grow as the technology evolves
- Avoid exports and imports
- Allow for copying settings between vehicles
- Place settings in normal use order, but allow free navigation
- Limit number of visible settings
- Use visualizations for feedback
- Explain difficult concepts

The results of the study performed showed that the new concept was easier to use for beginner users, and in general, generated a better user experience regarding emotional response as well. The concept was regarded as easier to use by beginner users. It was generally received well by expert users, but was thought to become more complex as more functionality was included.

In order to implement the solution, more functionality in terms of parameters need to be included, and the whole solution should be tested with both beginner users and expert users with all parameters included. However, we are confident that the user needs that are included in this report, as well as the design guidelines, represent both the main issues with the current KMA software and present remedies to those problems.

References

[1] 10 usability heuristics for user interface design. [Online; accessed May 8, 2021].

[2] Affinity diagram (or k-j method). [Online; accessed April 6, 2021].

[3] The history of kollmorgen. [Online; accessed May 25, 2021].

[4] How to generate thousands of ideas in mere minutes; the morphological matrix. [Online; accessed April 1, 2021].

[5] Interaction design foundation-design thinking. [Online; accessed June 15, 2021].

[6] Mobile tutorials: Wasted effort or efficiency boost? [Online; accessed May 8, 2021].

[7] Social sustainability. [Online; accessed May 21, 2021].

[8] Success rate: The simplest usability metric. [Online; accessed May 18, 2021].

[9] Un global - social sustainability. [Online; accessed May 21, 2021].

[10] Web content accessibility guidelines (wcag) 2.0. [Online; accessed May 18, 2021].

[11] Why use impact mapping? [Online; accessed March 25, 2021].

[12] Why you only need to test with 5 users. [Online; accessed May 8, 2021].

[13] ISO 9241-210:2010. Ergonomics of human-system interaction - part 210: Human-centred design for interactive systems (iso 9241-210:2010). *International Standards Organisation*, 2010, 2010.

[14] Joseph S. Dumas and Janice C. Redish. *A practical guide to usability testing*. Intellect Books, 1999.

[15] Ullrich Günter. *Automated guided vehicles, a primer with practical applications*. Springer Verlag Berlin, 2015.

[16] Klaus Scherer. What are emotions? and how can they be measured? *Social Science Information*, 44(4), 2005.

[17] Vera Shuman, Klaus Scherer, Johnny Fontaine, and Cristina Soriano. *The GRID meets the Wheel: Assessing emotional feeling via self-report*. 08 2015.

[18] Åsa Wikberg Nilsson, Åsa Ericson, and Peter Törlind. *Design: Process och metod*. Studentlitteratur AB, 2015.

A. Success rate metrics

Nr	Task description	Task (LD)	Success metric (LD)	Task (prototype 4)	Success metric (prototype 4)
1	Create a steer drive (SD) vehicle	Create an SD vehicle with * carrier type 1 * wheel base 1000 mm * length 1200 mm * width 400 mm * reference point 50 mm	*Full success:* Finds and enters every setting on the first try *Partial sucess*: Finds and enters every setting on the first or second try	Create a vehicle with * carrier type 1 * wheel base 1000 mm * length 1200 mm * width 400 mm * wheel configuration: SD 1 * reference point 50 mm	*Full success:* Finds and enters every setting on the first try *Partial sucess*: Finds and enters every setting on the first or second try
2	Create another SD vehicle	Create an SD vehicle with * carrier type 2 * wheel base 900 mm	*Full success:* Enters wheel base correctly, marks carrier type (CT) 2 and then unmarks CT 4 *Partial sucess:* Enters wheel base correctly, marks CT 2 but doesn't unmark CT 4 OR first unmarks CT 4 and then unmark all but CT 2	Create a vehicle with carrier type 2	*Full success:* Finds and enters every setting on the first try *Partial sucess*: Finds and enters every setting on the first or second try
3	Interpret vehicle and wheels visualization	*Same as task description*	*Full success:* Identifies vehicle and all wheels *Partial sucess:* Identifies vehicle and either SD wheel or back wheels	*Same as task description*	*Full success:* Identifies vehicles and all wheels *Partial sucess:* Identifies either SD wheel or back wheels
4	Create a diff drive vehicle	Create a diff vehicle with * carrier type 3 * gauge 300 mm	*Full success:* Finds and enters every setting on the first try *Partial sucess:* Needs hint for clicking "Next" to find diff vehicle OR enters some setting on the second try	Create a vehicle with * diff drive * left wheel y position 150 mm * right wheel y position 150 mm	*Full success:* Finds and enters every setting on the first try *Partial sucess*: Finds and enters every setting on the first or second try
5	Open a vehicle from another project	Import the vehicle found in the system StudySys and layout StudyLayout	*Full success:* Imports correctly at first attempt *Partial success:* Clicks wrong tab maximum 2 times OR receives hint that the vehicles in the layout are not listed	Find the menu "project explorer" and open "Vehicle X"	*Full success:* Finds project explorer and opens the vehicle on the first try *Partial sucess*: -II- second try
6	Add an antenna	Add an antenna with * Any name * X-coordinate 0 mm * Y-coordinate 100 mm * Length 100 mm * Width 3 mm	*Full success:* Finds and enters every setting on the first try *Partial sucess:* Finds and enters every setting on the first or second try	Add an antenna with * Any name * X-coordinate 0 mm * Y-coordinate 100 mm * Length 100 mm	*Full success:* Finds and enters every setting on the first try *Partial sucess:* Finds and enters every setting on the first or second try
7	Interpret antenna visualization	*Same as task description*	*Full success:* Identifies antenna	*Same as task description*	*Full success:* Identifies antenna
8	Move the antenna	Put the antenna on the other longside of the vehicle	*Full success:* Changes y-coordinate on the first try *Partial sucess:* -II- second try	Put the antenna on the other longside of the vehicle	*Full success:* Changes y-coordinate OR click-drags antenna on the first try *Partial sucess:* -II- second try
9	Delete the antenna	*Same as task description*	*Full success:* Clicks the delete-button of the interface on the first try *Partial sucess:* -II- second try	*Same as task description*	*Full success:* Clicks delete on keyboard OR right-clicks antenna and then delete on the first try *Partial success:* -II- second try
10	Navigate to safety sensors	Navigate to where you can configure the safety sensors	*Full success:* Navigates to "safety system setup" on the first try *Partial sucess:* -II- second try	Navigate to where you can configure the safety sensors	*Full success:* Navigates to "safety sensors" on the first try *Partial sucess:* -II- second try
11	Add a sensor	Add a sensor and give it a name	*Full success:* Clicks "new", then clicks name field, then writes name *Partial sucess:* Tries to write name without having clicked name field, then clicks name field and writes name	Add a sensor at the back of the vehicle	*Full success:* Navigates to "sensor configuration" then clicks "Back" on the first try *Partial sucess:* Finds everything on the first or second try
12	Move the sensor	Change the x-position of the sensor to 100 mm	*Full success:* Changes x-coordinate on the first try *Partial sucess:* -II- second try	Move the sensor to the front of the vehicle	*Full success:* Changes y-coordinate OR click-drags the sensor on the first try *Partial sucess:* Deletes sensor and adds a new one at the front of the vehicle
13	Find where you can add a safety field	Find where you can change the safety field of Sensor1	*Full success:* Double clicks "Sensor1" OR single clicks "Sensor1" and then clicks "Properties" on the first try *Partial sucess:* -II- second try	Find where you can add a safety field	*Full success:* Navigates to "safety fields" then clicks "Add new" on the first try *Partial sucess:* Finds everything on the first second try
14	Change the size of the safety field	Change the safety field "Nödstopp" from being 2000 mm long to being 3000 mm long	*Full success:* Changes both columns on the right *Partial sucess:* First clicks "Max Y" but doesn't enter a value, then finds and changes the columns on the right	Change the width and length of the safety field you just added	*Full success:* Finds the safety field and changes the values on the first try *Partial sucess:* Finds everything on the first or second try
15	Change the safety case max velocity	Change the safety system case "Långsamt" so that the max speed is 120 mm/s	*Full success:* Changes the max velocity on the first try *Partial sucess:* -II- second try	Add a new safety case and change the max speed to 120 mm/s	*Full success:* Finds safety case, then add new, and changes the value on the first try *Partial sucess:* Finds everything on the first or second try
16	Set the PLC bit 3 to "ON"	*Same as task description*	*Full success:* Clicks twice so the bit becomes "checked" *Partial sucess:* First clicks once so the bit becomes "filled", after a while clicks again so it becomes "checked"	*Same as task description*	*Full success:* Activates the bit, then clicks "on" OR clicks "on" on the first try *Partial sucess:* -II- second try
17	Activate the safety field	Set the safety field "Nödstopp" to active	*Full success:* Clicks add, then select fields, then checks "Nödstopp" on the first try *Partial sucess:* -II- second try	Activate the "Field 2" of "Sensor 1"	*Full success:* Selects Sensor1 and Field2, and clicks "Add" *Partial sucess:* Selects Sensor1 and Field2, receives hint that something more needs to be done and then clicks "Add"

B. Morphological matrix

	Referensenslösning	Funktion	Ide 1	Ide 2	Ide 3	Ide 4	Ide 5	Ide 6
	Referensenslösning	Funktion	Ide 1	Ide 2	Ide 3	Ide 4	Ide 5	Ide 6
1	Symbol	**Representera fordonet**	Som en skrinda (rätblock + hjul)	rätblock i olika fomrer	Ritning	Cirkelsektor (pizza-slice)	En häst	3D-rätblock
2	Symbol	**Representera säkerhetsfält**	Fält som man drar i för att ändra storlek. Kunna lägga till punkter för att ändra form	Visa hur safetyfieldet påverkar körvägarna med simulering (gif typ)	Välj safetyfields från bibliotek	Olika färg på warning och protection		
3	Bild på siten	**Påminna användare om verkligheten**	Webcam på site	Visualisera i olika nivåer (skiss, modell, 3d-modell i verklig miljö)	Vy med miljön där man kan klicka på olika delar (t ex klicka på golvet -> bromstestparametrar)	Interfacet byter till mer verkligt utseende (blixtrar till)	Kunderna MÅSTE svara på vissa frågor (t ex typ av golv)	Interface anpassat för användning på site
4	Catia-modell	**Medge custom truckrepresentation**	Bygga upp en truck av "legoklossar" i programmet	Stämplar av "truckformer" som kan kombineras	koppla en fysisk minitruck till programmet	välja bakgrund till fordonet	Dra och släppa olika delar för att bygga upp trucken	Olika "skins" på trucken
5	Google drive	**Inkorporering i större system**	Bild på en kropp, huvudet hur den tänker etc, alla program del av bilden	Visa kontraktet mellan kund och KMA, vad man har lovat att göra	En vy med en röd tråd med noder som är olika program i processen	Färger för de olika programmens faser	Olika simuleringar (layout, sensorer) pågår samtidigt i många små vyer	"Bin" som flyger från olika delar av programmet för att visa var man är i processen
6	Progressbar	**Process beskrivning**	Trucken blir "gladare" ju mer som är konfigurerat	Olika bakgrunds-färg beroende på vilken del som konfigureras	Visa ett värde baserat på hur mycket som är konfigurerat	Pie chart över hur långt man kommit och vad som är kvar att göra	Progressbar som i linkedin	Checklista
7	Chattfunktion	**Medge kommunikation mellan kollegor**	Möjliggöra "reaktioner" på ritningar i en gemensam miljö	Markera det som *inte* är relevant för vissa personer	Belöningssystem för att kommunicera mycket	Allt är online som Miro/ google	Dela-knapp	Hål/ plats där man kan lägga saker så att andra kan komma och titta
8	Printscreens	**Medge kommunikation externt**	Enkelt formulär i mail som servicetekniker kan fylla i så kommer det automatiskt in i programmet	Kunden/ partnern får automatiska "snapshot" uppdateringar	Ser/ får notis om förändringar utan att behöva öppna programmet	Inkorperera vanliga kommunikations plattformar så som teams och slack	Importera textfil, t ex datablad pdf, programmet identifierar och fyller själv i parametrar	Exportera pdf med hur det ser ut, kort beskrivning
9	Googla	**Söka efter funktioner**	Få förslag på liknande saker man sökt efter innan	"Andra sökte även efter"	Man hör var olika funktioner är genom att ropa på dem	Valla info, vallningsverktyg , drar över ett hav med info med ett verktyg som plockar/ synliggör	Utforska-verktyg - hovra över objekt för att visa förklaring	Överblick på fordonet som man kan klicka på för att komma till den konfigurationen
10	Använda gamla konfigurationer	**Medge användning av standardfordon**	Truckar rangordnas efter hur många funktioner de har	"Baksidan av en bok"	"Welcome to my truck"-video	Man kan jämföra två fordon bredvid varandra	Simulering med massa vagnar, man kan plocka upp en och lägga in den i sitt projekt	Ett fordon/template har ett fotavtryck
11	Historik över senast besökta hemsidor	**Arbeta iterativt**	Push-notiser "det var länge sedan du arbetade på den här modellen, vill du öppna nu?"	Meny som ändrar ordning så att det man gjorde för länge sedan ligger högst upp	Notis när man behöver uppdatera systemet pga ändringar hos kunden	Strukturerad projektdokumentation, drömdagbok	kvitto på vad som gjorts idag som kan sparas till nästa gång	Utgångsdatum på projekt
12	Manual	**Hjälpa användaren förstå vad som händer i programmet**	Highlighta det man har kvar att fylla i	Inställningarna uppdelade efter hårdvaran de berör	Snackbar	Checklista som uppdateras	Bokmärke	Stegvis, peka på nästa grej man ska fylla i
13	Steg-för-steg guide	**Förenkla inlärning**	Tutorial mode	Crash course vid uppdateringar	Advanced mode - visar fler inställningar	Olika modes, advanced, beginner, on site	Visa bara inställningar som är relevanta för den kunden	
14	Endast möjligt att inputa giltig data	**Förhindra fel**	Varningsbox "do you really want to do this?"	Siffror skrivs in i boxar där antalet är bestämt	Autosave-funktion	Historik där man kan backa till tidigare version	Kräv pinkod för "farliga" inställningar	Visualisera vad som förändras för varje inställning

	Referenslösning	**Funktion**	**Ide 7**	**8**	**9**	**10**	**11**	**12**
	Referenslösning	**Funktion**	**Ide 7**	**8**	**9**	**10**	**11**	**12**
1	Symbol	**Representera fordonet**	träbil	fysisk leksaksbil som man kopplar in	vridbara hjul	Animation över hur fordonet rör sig	Mörk färg på drivhjul för att behålla mental modell	3d-bild som går att rotera
2	Symbol	**Representera säkerhetsfält**						
3	Bild på siten	**Påminna användare om verkligheten**	Verklighetskikare	Filmer med t ex bromstestet	Lyssna på miljön medans man konfigurerar	Lager av "verklighet" som går att tända och släcka	Vy där man ser saker från vagnens perspektiv	"Remember your safety field must be .. in areas that are .."
4	Catia-modell	**Medge custom truckrepresentation**	man får önska sig något	Dra i hjul för att placera ut	Man kan skriva in siffror i bilden så uppdateras bilden	Fält som man drar i för att ändra storlek. Kunna lägga till punkter för att ändra form		
5	Google drive	**Inkorporering i större system**	En tråd mellan delar av projektet som i figma	Vy med alla program som upplagda kort på en dashboard	En "hylla" för de mest använda programmen	Karta över hur programmen hänger ihop		
6	Progressbar	**Process beskrivning**						
7	Chattfunktion	**Medge kommunikation mellan kollegor**	Olika språk	Lägga in polls för att rösta på koncept	Personifiera ens arbetsplats så att andra kan se vem som gjort vad	Meddelandekanon	Se om andra är inloggade	Möjlighet att "merga" templates
8	Printscreens	**Medge kommunikation externt**	Templates öppnas som pdf om man inte har programmet installerats					
9	Googla	**Söka efter funktioner**						
10	Använda gamla konfigurationer	**Medge användning av standardfordon**	"Bada" fordon i olika baljor av parametrar	Man kan välja ikon för fordonen	Varje standardfordon får ett smeknamn	Vikta egenskaper mot varandra	Man samlar en grupp med templates och så kan man se vilka egenskaper gruppen har	Flaggor för olika egenskaper
11	Historik över senast besökta hemsidor	**Arbeta iterativt**	Påminna om att ta raster/ byta projektdel	Lätt att "packa ner" filer att ta med till site	Det kommer upp spontant vad man kan göra idag. "Vad kan jag göra idag"-knapp	Projektdelar "möglar" när man inte är inne och kollar till dom		
12	Manual	**Hjälpa användaren förstå vad som händer i programmet**	Google maps i programmet "här är jag, hit vill jag"	Extern pryl som håller koll på projektet	Bläddra mellan sidor	Visualisera med lager	Röst som berättar vad som händer i programmet	Kalender
13	Steg-för-steg guide	**Förenkla inlärning**						
14	Endast möjligt att inputa giltig data	**Förhindra fel**	Highlighta förändringar					

	Referenslösning	**Funktion**	**13**	**14**	**15**	**16**	**17**	**18**
	Referenslösning	**Funktion**	**13**	**14**	**15**	**16**	**17**	**18**
1	Symbol	**Representera fordonet**	Foto på fordonet					
2	Symbol	**Representera säkerhetsfält**						
3	Bild på siten	**Påminna användare om verkligheten**						
4	Catia-modell	**Medge custom truckrepresentation**						
5	Google drive	**Inkorporering i större system**						
6	Progressbar	**Process beskrivning**						
7	Chattfunktion	**Medge kommunikation mellan kollegor**	Se en överblick på vad andra befinner sig (som Miro)	Varje användare har en "store" sär man kan lägga vissa funktioner/ egenskaper	"Batsignals" till kollegor	Ställer frågor i "etern" och får svar/filer/länkar som svar	Vissa standardfrågor om miljö/ trucktyp är mandatory att fylla i	Inkoorperera vanliga kommunikations plattformar som teams/ slack
8	Printscreens	**Medge kommunikation externt**						
9	Googla	**Söka efter funktioner**						
10	Använda gamla konfigurationer	**Medge användning av standardfordon**	Laddar upp fordonet -> ett fordon som kör	Allting har en signatur så man vet vem man ska fråga	Bibliotek i trädstruktur	Template-mode i vehicle config	Online-bibliotek med standardfordon	Filtrera efter funktioner
11	Historik över senast besökta hemsidor	**Arbeta iterativt**						
12	Manual	**Hjälpa användaren förstå vad som händer i programmet**	Gamla och nya filer har olika färg	Sy fast flikar i ett fönster så att de kan flyttas runt	Dashboard som man kan dra in fordon i (som google drive)	Vy som är en överblick över fordonet, klicka på olika delar för att komma till den inställningen		
13	Steg-för-steg guide	**Förenkla inlärning**						
14	Endast möjligt att inputa giltig data	**Förhindra fel**						

	Referensenslösning	**Funktion**	**19**	**20**
	Referensenslösning	**Funktion**	**19**	**20**
1	Symbol	**Representera fordonet**		
2	Symbol	**Representera säkerhetsfält**		
3	Bild på siten	**Påminna användare om verkligheten**		
4	Catia-modell	**Medge custom truckrepresentation**		
5	Google drive	**Inkorporering i större system**		
6	Progressbar	**Process beskrivning**		
7	Chattfunktion	**Medge kommunikation mellan kollegor**	Flera kan redigera samma fordon samtidigt	Share link: edit / view and comment
8	Printscreens	**Medge kommunikation externt**		
9	Googla	**Söka efter funktioner**		
10	Använda gamla konfigurationer	**Medge användning av standardfordon**		
11	Historik över senast besökta hemsidor	**Arbeta iterativt**		
12	Manual	**Hjälpa användaren förstå vad som händer i programmet**		
13	Steg-för-steg guide	**Förenkla inlärning**		
14	Endast möjligt att inputa giltig data	**Förhindra fel**		

C. Complete user needs list

Group	Need	Pugh matrix	Guidelines	Requirements	Importance			
Clarity	Information is presented in a clear way		x					
Clarity	The most important information is most visible		x					
Clarity	Images and visualizations are clear and serve a purpose to the given situation in which they are presented		x					
Clarity	Exclude functions that are never used			x				
Clarity	It is clear what actions lead to, and where input information end up		x					
Clarity	It is clear what input affects the system and what is only for visualization		x					
Clarity	The user receives feedback when making an input		x					
Clarity	The vehicle can be identified by physical type			x				
Clarity	Minimize risk of overwhelming amount of information		x					
Collaboration	Allow sharing and reuse of configurations		x					
Collaboration	Allow for collaboration within the software		x					
Collaboration	It is easy to share information from the solution with someone who doesn't have the program		x					
Communication	Allow for communication between KMA, customer and end user		x					
Consistency	The solution behaves in a consistent way throughout usage		x					
Consistency	Information is presented in a similar way throughout the usage		x					
Context	Help the user understand how the environment may affect the system in different warehouses		x					impact
Context	Visually and functionally a part of a bigger software universe			x				
Context	Allow changes further on in the process of creating a new AGV system		x					
Context	Visually be a part of the KMA design universe		x					
Context	Usable on site while interacting with an AGV		x					
Context	Make the user aware of the current state		x					
Errors	Allow automatic update of references to encoder speed etc (ask Michael more about this)			x				
Errors	Prevent use errors, slips and cognitive aware mistakes		x					
Errors	Allow for seing errors comitted		x					
Errors	Allow for fixing errors comitted		x					
Errors	Minimize risk of errors when starting from template/old vehicle config		x					
Errors	Allow tracing changes made		x					
Export/ Import	Minimize errors caused by exessive export/ import		x					deliverable
Export/import	Minimize the need of exporting		x					
Export/import	Possible to import safety fields			x				
Flexibility	Allow working with older vehicle types			x				
Flexibility	Allow for partners making custom, and standard vehicles		x					
Flexibility	Allow working with different types of vehicles		x					
Flexibility	Include a function that fills the need for using carrier types, a tool that allows the user to devide vehicles into diffenet groups			x				
Flexibility	Allow use of other PLC programming softwares			x				
Flexibility	Allow modular use of configuration sets		x					
Flexibility	Enable more PLC bits			x				
Flexibility	Allow easy copying between templates / old vehicles		x					
Information handling	The user only needs to input every information in one place			x				
Information handling	It is clear what information has been inputed		x					
Information handling	It is clear what data and calculations information is based on (when applicable)		x					
Users	Possible to do a vehicle configuration without a lot of experience		x					
Users	Ease usage for beginner users		x					
Users	Help expert users learn the new interface quickly, maintaining mental model		x					
Users	Allow for iterative work		x					
Function	The wheel position can be configured			x				

Group	Need	Pugh matrix	Guidelines	Requirements	Importance			
Clarity	Clarify what units are used			x				
Flexibility	Be able to change units used in the program			x				
Use experience	Decrease negative emotions generated by usage		x					
Use experience	Make the solution look modern or at least up to date		x					
Users	Enable common short commandoes (such as tab and double click)			x				
Flexibility	Enable updating multiple configurations simultaneously		x					
Collaboration	Allow documating information in text about the vehicle		x					
Clarity	It is clear what can be changed in the solution and what can't		x					
Clarity	Selection of words make sense		x					
Clarity	Icons and names of different parts of the studio must me coherent, but easy to separate from eachother		x					
Function	The safety fields can be configured			x				
Function	The sensor locations can be configured			x				
Function	The antenna locations can be configured			x				
Flexibility	It is easy to find what templates include what functionality		x					

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 20xx
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY