Graphical User Interface Design of a Maintenance Support System
Using Prototyping and User-Centred Design

Daniel Axelsson

Supervisor: Johan Blomkvist
Examiner: Stefan Holmlid
Supervisor at Saab AB: Johan Lignell
Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/her own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Daniel Axelsson
Abstract

The interest in the complex relationship between the behaviour of users and the design of interactive system has been significantly increased as the digital technology has advanced. This has led to usability becoming one of the key elements in user-centred interaction design. Systems need to be designed in a usable way; efficient, use-enhancing, flexible and learnable and the design should also meet the user’s needs and aspirations. This thesis aimed to develop a more usable prototype of the Maintenance Ground Support System (MGSS), using prototyping and a user-centred design approach. The prototype was developed using an adaptation on the evolutionary software development process that consisted of four iterative steps. The prototypes were created, tested and evaluated with surrogate and end-users. The design of the prototype is based on a customizable and simple dashboard application that supports multiple user needs and requirements, in a familiar environment where the user can feel confident and be in control. Based on usability testing, the prototype was concluded to be more efficient, understandable as well as easier to use than the existing system.

Keywords: Usability, Prototyping, Interaction Design, User-Centred Design, Design Evaluation
Acknowledgement

First, I would like to express my gratitude towards Saab AB for this thesis opportunity. A special thanks to my supervisor Johan Lignell, who has supported me and given me necessary knowledge I needed to complete this thesis. Furthermore, I would like to express my gratitude towards the participants of my focus groups at Saab AB for valuable feedback and interesting discussions throughout the design process. I would also like to express my gratitude towards the TDMG group for a warm welcome.

From Linköping University, I would like to thank my supervisor Johan Blomkvist for invaluable support and feedback throughout the thesis process. All your contributions and guidance has been greatly appreciated. Additionally, I would to express my gratitude towards my examiner Stefan Holmlid for sharing valuable knowledge.

Finally, I would like to thank my friends and family for all the love and support. A special thanks to my lunch and coffee colleagues Johan Agnvall and Jacob Hjelmqvist for the company and the laughs.

Linköping, 2018-06-11

Daniel Axelsson
# Table of Contents

**Copyright** ................................................................................................................................. ii  
**Abstract** ....................................................................................................................................... iii  
**Acknowledgement** ......................................................................................................................... v  
**Figures** ........................................................................................................................................... ix  
**Tables** ............................................................................................................................................ x  
1 **Introduction** ............................................................................................................................... 1  
1.1 Background ................................................................................................................................. 1  
1.1.1 Saab AB ...................................................................................................................................... 1  
1.1.2 The Maintenance Ground Support System .............................................................................. 1  
1.1.3 Previous work ......................................................................................................................... 2  
1.2 Aim and purpose .......................................................................................................................... 2  
1.3 Delimits and limitations ................................................................................................................. 2  
1.4 MGSS – Walkthrough .................................................................................................................. 3  
2 **Theoretical framework** ............................................................................................................... 9  
2.1 User-centred design ..................................................................................................................... 9  
2.1.1 Participatory Design ............................................................................................................... 9  
2.1.2 Design Exploration ............................................................................................................... 10  
2.2 Design Evaluation ....................................................................................................................... 11  
2.3 Human Needs and Aspirations .................................................................................................... 12  
3 **Pre-study** ..................................................................................................................................... 15  
3.1 User needs .................................................................................................................................... 15  
3.1.1 Previous work on user needs .................................................................................................. 15  
3.1.2 Complementary user needs .................................................................................................... 16  
3.2 Persona ......................................................................................................................................... 16  
3.3 Benchmarking ............................................................................................................................. 16  
3.3.1 SAS/STAT ............................................................................................................................... 16  
3.3.2 NVivo ....................................................................................................................................... 17  
3.3.3 MAXQDA ............................................................................................................................... 18  
3.3.4 Dedoose .................................................................................................................................. 18  
3.3.5 Adaptive Insights .................................................................................................................. 19  
3.3.6 Adaptive Vision ...................................................................................................................... 19  
3.4 Early Prototyping .......................................................................................................................... 20  
3.5 Workshop ..................................................................................................................................... 21  
3.5.1 Procedure ................................................................................................................................ 21  
3.5.2 Workshop Result ..................................................................................................................... 21  
4 **Design Process** ............................................................................................................................ 23  
4.1 Proof-of-context prototyping ...................................................................................................... 23  
4.1.1 Design ...................................................................................................................................... 23  
4.1.2 Testing ...................................................................................................................................... 28  
4.1.3 Evaluation ............................................................................................................................... 28  
4.2 Proof-of-concept prototyping ...................................................................................................... 29  
4.2.1 Implementation ...................................................................................................................... 29  
4.2.2 Design ...................................................................................................................................... 29  
4.2.3 Testing ...................................................................................................................................... 34  
4.2.4 Evaluation ................................................................................................................................ 34  
4.3 Proof-of-product prototyping ...................................................................................................... 36  
4.3.1 Implementation ...................................................................................................................... 36  
4.3.2 Design ...................................................................................................................................... 37  
4.3.3 Testing ...................................................................................................................................... 41
4.3.4 Evaluation ......................................................................................................................... 42
4.4 Proof-of-value prototyping .................................................................................................. 44
  4.4.1 Implementation .................................................................................................................. 44
  4.4.2 Design - Result ............................................................................................................... 44
  4.4.3 Testing ............................................................................................................................. 49
  4.4.4 Final evaluation .............................................................................................................. 50
5 Discussion ................................................................................................................................ 53
  5.1 Method .................................................................................................................................. 53
    5.1.1 Pre-study ......................................................................................................................... 53
    5.1.2 Design Process ............................................................................................................... 53
  5.2 Results .................................................................................................................................. 55
    5.2.1 Graphical user interface design ....................................................................................... 55
    5.2.2 Evaluation ...................................................................................................................... 56
6 Conclusion ................................................................................................................................ 59
  6.1 Future research ...................................................................................................................... 59
7 Reference List ........................................................................................................................... 60
8 Appendix .................................................................................................................................... 64
  8.1 Appendix 1 – Persona ......................................................................................................... 64
Figures

Figure 1.1 – Cycle Overview Window, DCO, the main window of MGSS .................................................. 3
Figure 1.2 – Download Package Repository window, DPR ........................................................................ 4
Figure 1.3 – Add Standard Cycle window .................................................................................................. 4
Figure 1.4 – Cycle Information window, DCI .......................................................................................... 5
Figure 1.5 – The Fault Reports tab in the DCI window ............................................................................. 6
Figure 1.6 – The Manual Fault Isolation window .................................................................................... 6
Figure 1.7 – An example of an export window .......................................................................................... 7
Figure 3.1 – An example of a SAS/STAT project with a main window and 3 data visualizations ............. 17
Figure 3.2 – An example of an NVivo workspace .................................................................................... 17
Figure 3.3 – An example of the user interface of MAXQDA with user customized windows ................. 18
Figure 3.4 – An example on a Dedoose project ........................................................................................ 18
Figure 3.5 – An example of a spreadsheet-like budgeting window in Adaptive Insights ....................... 19
Figure 3.6 – An example of how different tools are used to analyse objects in Adaptive Vision ........... 19
Figure 3.7 – Examples of early prototypes produced to facilitate communication ............................... 20
Figure 4.1 – Graphical representation of the design process used to develop the prototype .............. 23
Figure 4.2 – Same, Same, but Different .................................................................................................... 26
Figure 4.3 – Command System .............................................................................................................. 26
Figure 4.4 – Module Based ....................................................................................................................... 26
Figure 4.5 – Adaptive Software ............................................................................................................... 26
Figure 4.6 – Contextual Application ........................................................................................................ 26
Figure 4.7 – Split Vision .......................................................................................................................... 27
Figure 4.8 – Simply Advanced ................................................................................................................ 27
Figure 4.9 – Advanced + Simple ............................................................................................................. 27
Figure 4.10 – Dashboard ........................................................................................................................... 27
Figure 4.11 – Export Focus ....................................................................................................................... 27
Figure 4.12 – The general elements used to complement the concept prototypes .................................. 30
Figure 4.13 – Examples of interactive elements used in the Clean Setup concept ............................... 32
Figure 4.14 – Examples of interactive elements used in the Adaptive Dashboard concept .................. 32
Figure 4.15 – Examples of interactive elements used in the Tab-board concept .................................. 32
Figure 4.16 – Examples of interactive elements used in the Same, Same, but Dashboard concept ....... 33
Figure 4.17 – Examples of interactive elements used in the Activity Dashboard concept .................... 33
Figure 4.18 – Representation of the fundamental aspects of the co-created concept ............................. 36
Figure 4.19 – The starting view of the proof-of-product prototype ......................................................... 37
Figure 4.20 – The main view of the prototype when cycles have been imported ................................ 38
Figure 4.21 – The Fault Report view with four different tabs ................................................................. 39
Figure 4.22 – The main view with when all items from the analyse list have been confirmed ............. 40
Figure 4.23 – Example of the export view as well as the notification panel .......................................... 41
Figure 4.24 – Examples of changes that was implemented after the pilot test ..................................... 41
Figure 4.25 – The starting view of the final prototype ............................................................................. 45
Figure 4.26 – The notification panel and import window of the prototype ........................................... 46
Figure 4.27 – The main view of the GUI with numerous cycles imported from the DTU .................... 47
Figure 4.28 – Analyzing a cycle using the prototype .............................................................................. 47
Figure 4.29 – The fault reports tab of the prototype .............................................................................. 48
Figure 4.30 – The function explorer in the All functions tool ................................................................. 49
Figure 4.31 – Mean SUS score of the prototype and MGSS. Error bars represent 95% CI .................... 51
Tables

Table 2.1 – The NADI-model: four layers of human Needs and Aspirations for a Design and Innovation process. (Van Der Bijl-Brouwer & Dorst, 2014) .............................................................. 12
Table 4.1 – Distribution of participants’ votes ........................................................................ 28
Table 4.2 – Amount of errors and distribution of participants’ votes per concept ................ 35
Table 4.3 – Concluded changes to be implemented in the final prototype ................................ 43
Table 4.4 – Mean user actions per task using the prototype and MGSS ................................... 51
1 Introduction

Some of the key design issues to be concerned in interaction design are reliability, efficiency and function (Candy & Costello, 2008). By only focusing on these design issues when designing interactive systems, poorly designed systems which are difficult to operate and complicated to learn often emerge. Additionally, these systems tend to be under- or poorly used and frustrating for users (Weichbroth & Sikorski, 2015), leading to a costly outcome for the organization using the system (Maguire, 2001). With a user-centred design process the amount of user errors can be reduced and the overall productivity can be improved (Maguire, 2001).

The interest in the complex relationship between the behaviour of users and the design of interactive system has been significantly increased as the digital technology has advanced (Candy & Costello, 2008). This has led to usability becoming one of the key elements in interaction design. Systems need to be designed in a usable way; efficient, use-enhancing, flexible and learnable and the design should also meet the user’s needs and aspirations (Van der Bijl-Brouwer & Dorst, 2017; Shackel, 1986). Furthermore, it is important that interactive systems and graphical user interfaces (GUIs) are designed for the users by involving the users. This makes the outcome predictable, by allowing users to evaluate upon the design (Gasparic, Janes, Ricci, Murphy, & Gurbanov, 2017). Additionally, the system becomes more flexible, accessible and adaptable (Hertzum & Simonsen, 2012).

One of the key elements in successful design and innovation is to gain user insights (Martin, 2009) and knowledge (Criscitelli & Goodwin, 2017). If the concerns, motives, goals and the environment of the users are well understood the innovative process can be improved (Hekkert & van Dijk, 2011). By involving users, the goal is to develop more usable systems by increasing the knowledge of the users’ needs and goals, which reduces the risk of late and costly changes (House & Hill, 1997). This improves productivity, safety, performance and reduce the need for support and training (Österman, 2012; Norman, 2013).

1.1 Background

In this section, Saab AB and The Maintenance Ground Support System, MGSS, will be introduced in brief, followed by related work to this thesis. A more thorough walkthrough of MGSS’s graphical user interface will be presented in chapter 1.4 MGSS – Walkthrough.

1.1.1 Saab AB

Saab is a global military defence and civil security company who operates on every continent. Saab’s most famous product is their aircraft Gripen. This aircraft needs to be operating at a low cost and with high sustainability to be effective (Saab AB, 2018). To be able to do this, knowledge about the wear of the aircraft in and before flight is crucial. Gripen is equipped with a Dara Transfer Unit (DTU) to deal with this task. The DTU can log thousands of different data points from every single component in the aircraft. MGSS is a maintenance system developed by Saab, making it possible to monitor and analyse the wellbeing of an aircraft, based on these data.

1.1.2 The Maintenance Ground Support System

The MGSS is used to export data from the aircraft to other systems, collect and present operational data, support maintenance of aircraft and flights and to troubleshoot fault events. MGSS is used after each flight, but can be used before flights to run diagnostics and to update new software.

According to the development team of MGSS, the system is an outdated system as far as interaction and usability is concerned. The system has been designed concerning the key design issues described by Candy & Costello (2008). Over the years the system has been updated with more and better
functions to widen the system’s ability to process and display flight data, by request of the customer. Throughout this functional improvement, the interface design has not been prioritized. This has made MGSS a system that’s able to present enormous amounts of data, but not in a satisfactory and efficient way (Saab AB, 2018). A system that provides a large amount of functions makes the system overwhelming for the user (Gasparic, Janes, Ricci, Murphy, & Gurbanov, 2017). Additionally, these functional improvements have made the graphical user interface’s interaction levels deep, making them incomprehensible, hard to learn and understand. By adding functions without concerning the design of the user interface, the interface has become confusing and inconsistent, making it harder for users to make decisions based on the information the system holds (Saffer, 2010).

The system is primarily used by expert aircraft technicians. The technicians are not highly trained computer analysts. According to the developers of the MGSS, the technicians are not used to handling computerized, analytical systems.

1.1.3 Previous work
During the spring of 2016 a thesis report was conducted, examining how MGSS could be designed using the activity checklist combined with the sequence model (Magnusson, 2016). Magnusson (2016) raised deficiencies and qualities of MGSS as well as explained the users’ behaviours and needs. Based on this a prototype was designed and evaluated with surrogate users. The result showed that the prototype had improved usability. This thesis will take these results into consideration when a persona was created as means of representing the users’ needs and aspirations.

1.2 Aim and purpose
The aim of this thesis is to develop a mock-up prototype of MGSS designed for the users, making the system more usable in terms of efficiency, use-enhancement, flexibility and learnability (Van der Bijl-Brouwer & Dorst, 2017; Shackel, 1986). This thesis will demonstrate how prototyping and a user-centred design approach can be used to improve the usability of MGSS.

Based on the operational definition of usability this thesis will address the following question:

- How can design improve the usability of MGSS?
  - How can MGSS be designed to improve the efficiency of analytical tasks?
  - How can MGSS be designed to enhance usage?
  - How can MGSS be designed to increase flexibility?
  - How can MGSS be designed to improve learnability?

Efficiency is here referred to as the users’ ability to perform familiar tasks with fewer actions than before. Whereas learnability is the user’s ability to quickly understand and get familiar with the system, i.e. adapting to processes, sequences, functions and tools. Use-enhancement is referred to characteristics making the graphical user interface simple and easy to use, from a users’ perspective. In this context, flexibility is referred to as the user’s ability to modify and customize specific actions and processes in the system as well as being able to perform actions in several different ways.

1.3 Delimits and limitations
MGSS is a highly complex system and due to the 20-week time limit of this thesis, prototyping a system which demonstrates every user case with the functions and/or tools needed was not possible. After consulting with the supervisor from Saab AB the prototyping sessions as well as the final prototype only considered the primary and most frequently used functions of MGSS. Additionally, there was a limited possibility to integrate primary end-users and difficult to involve the customer, Försvarets
materielverk (FMV), in the design process. Therefore, the stakeholders that participated in the design process were the developers of MGSS and one end-users; a flight test engineer.

1.4 MGSS – Walkthrough

In this section the user interface of MGSS will be presented and explained. The user interface contains secret information and therefore, the pictures along with respective data presented in this section have been altered by request of Saab AB.

MGSS is mainly used to export operational data recorded from various aircraft components, such as the engine or the APU (auxiliary power unit), to other systems. MGSS is also used for importing, visualizing, troubleshooting and deep analysis of various data recorded from an aircraft. The MGSS graphical user interface Microsoft .Net framework, using WinForms as the GUI library and C# as the programming language. The GUI resembles a traditional, grid based windows application in that new windows containing information opens every time a function or tool is used. Additionally, the flexibility and the assistance of the GUI is limited.

When starting MGSS, the user is met by a large grid window called Cycle Overview (DCO). At first, this DCO is empty. The user needs to import data or download a package repository and then filter out preferred information. Subsequently, the user can hit the refresh button to fill the grid with information. MGSS now present information of all imported cycles, as seen in Figure 1.1. The grid has 19 columns which represents information such as aircraft, cycle type and fault events. The number of rows in this DCO is dependent on the amount of cycles imported. A cycle is here referred to as a series of events occurring during a specific time frame. The “fault events” column contains all events that occurred during the specific cycle, but this doesn’t necessarily mean that those events are critical. The user needs to analyse this specific cycle to interpret the criticality of the event. At the top of the GUI a ribbon can be found containing the name of the current window together with minimize, restore down and close functions. Below the ribbon a main tool bar can be found, and like other windows applications designed in the 1990’s, consisting of text based drop down menus such as File, Edit, View and Tools. In the upper left corner, below the main tool bar, a few icons can be found, representing some of the frequently used tools and functions. On the left-hand side of the window the filtering view that was mentioned earlier can be seen.

![Figure 1.1 – Cycle Overview Window, DCO, the main window of MGSS](image-url)
Described by employees at Saab AB, there are some more frequently used tools and functions in MGSS. These are Download Package Repository, Import, Add Standard Cycle, Cycle Information (DCI), Manual Fault Isolation (MFI) and Export. These functions are crucial to the daily usage of MGSS (Magnusson, 2016). These functions and tools are described below.

To be able to analyse data in MGSS the user needs to import the data from the DTU to the system. This can be done by using the Download Package Repository window, that can be accessed from the DCO, by using the icons in the upper left area. Here, the user can find, show and unpack data points from the DTU, i.e. create cycle data. The Download Package File Filter view to the left lets the user filter out specific file types from the data, e.g. data from a specific aircraft, Figure 1.2. In the bottom left area, the function Import Download Packages opens yet another window where the user can browse for .zip files, import and subsequently download files to MGSS, i.e. adding the cycles to the DCO grid. The Import window opens, looks and behaves much like the Download Package Repository window.

![Figure 1.2 – Download Package Repository window, DPR](image)

Every cycle that has been logged on a DTU has an attached log sheet, containing information from the cycle. If the DTU did not log a cycle the user must create a cycle manually based on the information on the log sheet. The function for handling this task is called Add Standard Cycle, Figure 1.3.

![Figure 1.3 – Add Standard Cycle window](image)
The Cycle Information (DCI) window is opened from the DCO by e.g. double-clicking a specific cycle. This window shows data related to the specific cycle and allows the user to edit certain data, Figure 1.4. As with the DCO, the window contains a ribbon, a tool bar with text based drop down menus and some icons representing specific functions. The icons here are not the same as in the DCO, instead they represent the most frequently used tools for this specific type window. This window includes 9 tabs, each showing different types of cycle information. The event tab, presented in the figure is the default tab when opening the DCI. In this tab, the user can see information concerning the elapsed time before the event happened, the event number and a name related to the fault event. Additionally, the user gets an indication of the criticality of the fault event along with what type of fault event that has occurred and to what materiel group (MG) the event corresponds. In this window, the experienced user can occasionally interpret the fault event directly based on the event number.

![Cycle Information window, DCI](image)

Figure 1.4 – Cycle Information window, DCI

But more commonly the user needs to analyse an unfamiliar fault event to get an explanation. This is often done by opening the tab “Fault Report”, Figure 1.5, in the DCI window. In this window, to the left, the user is presented with the same information as in the previous tab. To the right, the user gets three more tabs containing additional information of the Fault Report, e.g. codes as shown in Figure 1.5. To interpret these codes the user uses an external document called Aircraft Maintenance Publication (AMP). MGSS still does not indicate how critical an event is, nor why the event occurred. The only way for the user to get this information is to compare the codes given in this window to the information in the AMP.
The Manual Fault Isolation (MFI) window is used when additional troubleshooting – more than the analysis possible in the DCI – needs to be done. This window can be opened in several ways; by right-clicking a specific cycle and choose the function from a drop-down menu, by using hotkeys and by using the icons in the upper left area of the DCI. To the left in this window the GUI presents all the variables, stored by the DTU, in a specific cycle, Figure 1.6. To the right of this, the MFI uses graphical elements such as coloured shapes and lines to represent events and specific variables in a cycle. Worth noting is that the specific shapes and colours does not in any way indicate the criticality of the events or variables, nor the type of event that has occurred.
MGSS contain of several different Export functions, which are all used. Fundamentally, all the export functions have the same functionality and design; a grid showing the selected cycles with respective information, an export folder field and an export button as exemplified in Figure 1.7.

![Figure 1.7 – An example of an export window](image)

As mentioned earlier, MGSS holds a vast number of functions and tools that sometimes serves a purpose in the analysing and troubleshooting process. This correlates to the fact that the customer constantly requests functional system updates to widen the system’s ability to process and display flight data.
2 Theoretical framework

This section explains the theoretical framework that is needed to understand the contents of this thesis.

2.1 User-centred design

One of the most beneficial way to generate ideas, to be more creative and to produce innovative designs is to bring people with different backgrounds and training together in multidisciplinary teams to work jointly in the design process (Preece, Rogers, & Sharp, 2002; Humphreys, Leung, & Weakley, 2008). Specifically, at the early stages of a design process it is hard to develop design specifications – what, how, where and why to design a specific artefact – due to the uncertain nature of a design process (Paternò, 2013). It is therefore important to gather user information and get feedback on the design process at the early stages, and subsequently to retain this user-collaboration throughout the design process. When successful in doing so, the understanding of the users is improved which subsequently makes it possible to create more acceptable and usable systems (Lindgaard, et al., 2006; Wilkinson, 2014).

When designing solutions for complex problems more knowledge than a single person possesses is needed (Cooper, 2004). By actively involving users with relevant knowledge of the environment and context in which the design solution will be used, the commitment and acceptance to the new solution will be enhanced (Maguire, 2001). Collaboration is considered a key feature in end-user development (Humphreys, Leung, & Weakley, 2008). If the users are technical experts in the specific development area, and therefore possesses similar knowledge to the designer, the potential of collaboration within the design process is improved (Sharp, Rogers, & Preece, 2007). However, end-users tend to lack the specific design training and knowledge on development approaches are therefore not useful when collaborating with end-users in a development process (Paternò, 2013).

Collaborative development can be done with experienced users as well as novices. The major difference is that experienced users have a deeper understanding of the many problems in a specific domain and what solutions that are suited for the specific problems (Shute, 2012). An experienced user can mentally stand back from the details and specifics of a system, making it possible to focus on the underlying principles instead of problem features that occur on the surface of the system (Ho, 2001; Cross, 2004). Furthermore, experienced users are thought to have the capacity to store and access information in larger cognitive chunks, compared to novices (Dorner, 1999; Nigel, 2004). Regardless of specific knowledge, user-centred design should consider all potential users’ experience, characteristics, abilities limitations in order enhance usability (Wilkinson, 2014). Additionally, when actively collaborate on design changes the users get empowered, feel engaged and have a sense of ownership, improving the innovative process (Hekkert & van Dijk, 2011; Criscitelli & Goodwin, 2017; Kanga, Choob, & Wattersc, 2015).

There are clear advantages to having end-users or experienced users involved in a design development process. However, it can be even more beneficial to involve a user representative with knowledge for design in the development process (Humphreys, Leung, & Weakley, 2008). The representative can translate design decisions to the end user, preparing users for change, whilst fostering new needs and ideas making the final solution more easily adoptable (Candy & Costello, 2008).

2.1.1 Participatory Design

Participatory design is a design approach to systems design, which emphasizes the cooperative process with both designers and users. This approach can be used to meet needs and goals of users successfully, by involving all potential users in design process (Kanga, Choob, & Wattersc, 2015). Halskov and Brodersen (2015) has formulated five fundamental aspects of participatory design:
Theoretical framework

User-centred design

- Politics – Everyone who is affected by a design decision should be able to influence it
- People – By being experts in their own working situations, people play a critical role in design
- Context – The fundamental starting point of the design process is the use situation
- Methods – Users can communicate and influence in the design process through methods
- Product – Participatory design aims to innovate, improving the quality of life

Participatory design aims to improve the quality of working life by better involving and consequently understanding real users (Wilkinson, 2014; Halskov & Brodersen, 2015). Additionally, participatory design envision use before actual use, a way to meet the complicated challenge of fully anticipating the actual use that is happening in the users’ lives (Ehn, 2008).

Participatory design focuses on verbal exploration of ideas and concepts in design, which is especially significant in the early stages of a design process (Hevner, March, Park, & Ram, 2004). This makes the use of verbal exploration appropriate when a designer is either designing or researching new innovations (Kanga, Choob, & Wattersc, 2015). Additionally, the close participation of users throughout the design process progresses the process itself through needs analysis, development of prototypes and evaluation (Sharma, et al., 2008).

Because of the complex nature of the design process of systems, feedback at early stages in the design process is necessary (Leggett & Bilda, 2008). The design process is rarely constrained to the limits of the project, meaning that the approach and outcome of a specific process is open. This makes it important to understand how a specific design can relate to users’ needs and how a design can meet these needs (Ehn, 2008). If end-users are involved in the design process and with whom the design is continuously and jointly evaluated the relationship between users’ needs and specific design solutions can be discovered (Weichbroth & Sikorski, 2015).

2.1.2 Design Exploration

Prototyping is one way of exploring and communicating different design solution and an effective method to evaluate designs at early stages of a design process (Bhatti, Brémond, Jessel, & Dang, 2015; Yang & Epstein, 2005). There are however different classifications of prototypes, usually used with specific goals, in different ways and stages in the design process: throwaway or evolutionary prototypes (Sommerville, 1995). Throwaway prototypes are usually used in the early stages of a design process to help understand and clarify user requirements. Subsequently, the evolutionary prototypes evolve through several iterative stages of designing and evaluation (Van der Bijl-Brouwer & Dorst, 2017; Markus, Majchrzak, & Gasser, 2002).

Before beginning the design process of an interface, it is important to understand the context of use and who would be using it (Humphreys, Leung, & Weakley, 2008). Different types of prototypes can be used to understand and explore what approach to take, to meet the users’ needs and the domain environment (Yang & Epstein, 2005). There are four different types of prototypes that can be used in a product design process (Ullman, 2003). They are based on their respective functionalities and suggested time of implementation. By joining the product design prototypes described by Ullman (2003) with prototypes described in a user-centred process (ISO, 2010), the different prototypes can be described as:
• A *proof-of-context* prototype is used in the initial stages of a design process to explore and understand context of use related to user requirements.
• A *proof-of-concept* prototype is usually used after the first prototype to explore design functionalities to meet the user requirements.
• A *proof-of-product* prototype should clarify the design and suggest implementation methods of the final solution.
• A *proof-of-value* prototype evaluates the design solution to examine if the design successfully meets the user requirements.

The *proof-of-context* and *proof-of-concept* prototypes should illustrate the functionalities and qualities of a design idea whilst motivating users to look at the problem in different way. However, the *proof-of-concept prototypes should do this in an interactive way*. The *proof-of-product* prototype should demonstrate interaction and usage and the *proof-of-value* prototype should in a structured way explore the usability of the design. The prototyping process is most effective if done in an iterative manner (Weichbroth & Sikorski, 2015) and the prototypes that are developed should illustrate some, but not all, features of an intended design (Kesseler & Knapen, 2006).

The process of prototyping a user interface is an approach to facilitate the otherwise complicated communication processes between designers and other stakeholders, such as users and/or customers. (Weichbroth & Sikorski, 2015) A user interface prototype creates a foundation on which a discussion easily can be held, making it possible for all stakeholders to learn more about the product and its requirements and subsequently to jointly explore different design solutions (Sikorski, 2012). Furthermore, user interface prototyping is beneficial in that it makes identification and validation of user requirements possible at early stages of a design process; it gives users a sense of ownership, improving the users’ attitude towards change; and frequent prototyping with a frequent contact with users lead to a more usable product, needing fewer corrections at the end of the design process (Weichbroth & Sikorski, 2015).

The development of complex software systems is often associated with high risk and costs (Rudd & Isensee, 1994). Prototyping is a technique used to reduce the design risk involved in the development process, making late changes less likely to occur, which leads to an overall reduction of the time and cost of the development process (House & Hill, 1997).

### 2.2 Design Evaluation

In the context of this thesis, *design exploration* refers to the process of creating different prototypes, whereas *design evaluation* refers to the testing and evaluation processes involving users.

The feedback information that evaluation of a design generates gives a better understanding of the problem, improves the product and the design process (Markus, Majchrzak, & Gasser, 2002). It is important to not only evolve the product design but also the design process based on the feedback given during evaluation with users (Hevner, March, Park, & Ram, 2004).

The consideration of users’ context of use, environment of interaction and prior experience are key factors in creating a usable and comprehensive user interface. Additionally, these factors are critical to consider when usability of a given product or system is evaluated (Nielsen J., 1993). One of the most important activities in user interface development is to evaluate the design throughout the process (Maguire, 2001). By evaluating design, user needs and/or requirements can more easily be met and informal expectations can be validated and confirmed (Weichbroth & Sikorski, 2015). Additionally, important feedback can continuously be gathered from users to refine the design (Maguire, 2001).
One of the key elements in a software development process is to identify and solve usability related problems as early as possible, preferably using the less formal and thus most cost-efficient methods of evaluation (Maguire, 2001). Maguire (2001) describes three different levels of formality when conducting design evaluation:

- The least formal, Participative evaluation;
- The intermediate, Assisted evaluation;
- The most formal, Controlled evaluation.

The least formal approach aims to understand how the user is thinking by conducting questioning. Focus is here to investigate the user’s impression of a set of designs; what the user thinks specific elements do and what the expected results are when performing an action. In Participative evaluation the users should be asked to suggest improvements. The intermediate level of formality is appropriate when the designer needs user feedback in a context related environment, as realistic as possible. The users in the Assisted evaluation are often asked to perform specific tasks whilst thinking out loud, with some assistance from the conductor if problems occur. To find out how successful users truly will be using the suggested user interface; the most formal Controlled evaluation is needed. This form of evaluation is supposed to be conducted in a replicated environment of the real working environment, with minimum assistance from the conductor (Maguire, 2001). When conducting usability testing or studies, both major and minor problems concerning the systems’ usability will occur.

Design evaluation is done either with experienced users or novices, depending on the expected outcome of the evaluation. Expert evaluation is an easy way to get feedback and recommendations based on several years of usage. However, since experienced users tend to have personal biases toward specific design features whereas novices are more likely to reveal genuine problems on the surface of a design (Maguire, 2001).

### 2.3 Human Needs and Aspirations

There are four levels of aspirations and needs that needs to be considered in a design and/or innovative process (Van Der Bijl-Brouwer & Dorst, 2014). The model developed by Van der Bijl-Brouwer and Dorst differs from a product related models such as (Hekkert & van Dijk, 2011) in that it can be used to identify and create deep human insight within any user-centred design process. The model is called Needs and Aspirations for application in a Design and Innovation process, hereafter called by the acronym NADI. This model focuses on including desires and ambitions, rather than just direct needs, hence the term “... and Aspirations”. The NADI-model consists of four levels of needs and aspirations: solutions, scenarios, goals and themes (Table 2.1).

<table>
<thead>
<tr>
<th>Solutions</th>
<th>What do people want or need? Which products, services or interventions do people want or need?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenarios</td>
<td>How do people want or need to interact with the solution in the context of use?</td>
</tr>
<tr>
<td>Goals</td>
<td>Why do people want to interact or behave in a certain way? What do they want to achieve within the context of the problem?</td>
</tr>
<tr>
<td>Themes</td>
<td>What is the underlying structure of the experience? What are their meanings and values outside the direct context of the problem?</td>
</tr>
</tbody>
</table>

The first level, Solutions, refers to characteristics of solutions to meet specific user needs, such as products and services. The second level, Scenarios, describes the relationship between context of use and user, in terms of interaction. The third and fourth level, Goals and Themes, describe why users
want or need a certain solution and scenario respectively. The difference in the two levels is that Goals describe what users want to accomplish within the context of a specific design problem, whereas Themes describes the users’ needs and aspirations independently of the context described in Goals. The needs in the first level needs to contribute to the needs in the second level, etcetera (Van Der Bijl-Brouwer & Dorst, 2014). With a deep understanding of the Themes within the NADI-model the designer can create radically new solutions that fulfil hidden needs of the users, making the solution more adoptable and successful (Verganti, 2016) By asking how and why, context independent user needs and values can be identified and then related to Themes (Van Der Bijl-Brouwer & Dorst, 2014).
3 Pre-study

Prior to the pre-study conducted by the author of this thesis, earlier design work on MGSS (Magnusson, 2016) as well as in an interview study to facilitate flight missions (Johansson, 2018) were studied to analyse user needs and requirements. In the pre-study of this thesis, a semi-structured interview with a flight test engineer at Saab AB was conducted to expand the knowledge on the users’ needs. Based on the result of the user studies from both the previous and the supplementary work, applied to the NADI-model, a persona (Cooper, 2004) was created as means of representing the users of MGSS. By request of Saab AB, a benchmarking study was conducted. The aim was to investigate software and applications capable of analysing quantitative and/or qualitative to be used as source of inspiration in the early prototyping process. To better understand context of use and user requirements a workshop was conducted at Saab AB.

3.1 User needs

To successfully design a graphical user interface that effectively meet end-user requirements, it is important to understand the needs of the user (Criscitelli & Goodwin, 2017). Understanding how and which functions to implement in a user interface is crucial to meet these needs (Allen, 1996). Additionally, a comprehensive understanding of the users’ work and the needs as well as difficulties associated with this creates a foundation on which to base design decisions (Wilkinson, 2014).

3.1.1 Previous work on user needs

During the spring of 2016 a master thesis report was conducted, examining how MGSS could be designed using the activity checklist combined with the sequence model (Magnusson, 2016). During the observation of an experienced user in the daily usage of MGSS, Magnusson (2016) identified key functions and goals within MGSS. The users’ main goals could be simplified as transferring data into the system, making a fault analysis and then exporting the data (Magnusson, 2016).

To better understand the users’ daily work with MGSS – and how this work relates to the users’ main goals – the users’ work flow was organized into sub goals by Magnusson (2016). This was done by using the activity checklist, and then visualized using the sequence model. The workflow of the users was then described by Magnusson (2016) as three main procedures: importing data from DTU, analysing a cycle and troubleshooting unfamiliar fault event.

Magnusson (2016) presented six major user needs, by combining knowledge insights from the activity checklist and sequence model. The needs are stated as follows: “Have the ability to see information from one specific aircraft, clearly see fault events from a cycle, get a simple overview, export of data through different exports, get feedback on which exports has been done and when and being able to create a cycle manually” (Magnusson, 2016, p. 25).

During the spring of 2018 a bachelor thesis was conducted as an interview study to facilitate flight missions by investigating users’ requirements on supports systems at Saab AB (Johansson, 2018). In five semi-structured interviews conducted at Saab AB, users who had experience with planning flight missions were interviewed. The interviewees included pilots, former pilots as well as support system planners and evaluators. The several user requirements and demands on supports systems at Saab AB, identified by Johansson (2018) included a system that is simple and easily accessible, a system that is easy to understand and use, transparency in automated processes within the system, a system that is flexible and intuitive use as well as a trustworthy system.
3.1.2 **Complementary user needs**

Additional user needs were identified in the semi-structured interview with the flight test engineer at Saab AB. The interview was conducted at the engineer’s office at Saab AB, for one and a half hours. The aim of the interview was to identify the users’ needs in the daily usage of MGSS. During the interview, notes were taken on A5-sized ruled paper. After the interview, the notes were confirmed by the interviewee by asking if there were any misinterpretations. The needs that was described by the interviewee were numerous and some too detailed and specific to be included in this thesis, due to the time limitations of this thesis. However, numerous major needs were identified to be included in representation of the user of MGSS.

The user needs to:

- get assistance and feedback when interpreting fault codes in MGSS;
- have better access to information and facts related to fault codes and fault events;
- work efficiently in a smart user interface that is not time-consuming;
- only be presented with essential information.

Based on the user needs of the previous as well as the supplementary work, applied to the NADI-model, a persona was created.

### 3.2 Persona

Because an end-user representative was problematic to include in the throughout the entirety of the design process, a persona was made as means of representing the users’ needs (Cooper, 2004). The persona presents the user Rick Johnsson, a flight test engineer at Saab AB, Appendix 1 – Persona. Based on the users’ needs and goals, the personas solutions, scenarios, goals and expectations were described. By analysing the solutions and scenarios the Themes, as described by Van Der Bijl-Brouwer and Dorst (2014), related to the daily work of the persona could be defined. By identifying and understanding these Themes, new solutions that fulfil deep hidden user needs could be explored (Verganti, 2016). The Themes of the persona were defined as "When working with MGSS Rick wants to share information and have control when doing so. But Rick needs assistance.”.

### 3.3 Benchmarking

By request of Saab AB, a benchmarking study was conducted. The aim was to investigate software and applications capable of analysing quantitative and/or qualitative data to be used as a source of inspiration for future studies as well as the early prototyping work in this thesis. In the following sections, a selection of software that were used as design inspiration are presented.

#### 3.3.1 SAS/STAT

SAS/STAT is a software developed to perform advances analytics, data management and statistical analytics. An example of an application of this software can be seen in Figure 3.1. The software includes exact techniques, statistical modelling and modern methods for analysing data making all kinds of data set analysable (SAS, 2018). SAS/STAT contains a vast number of built-in and customizable features for data visualization, e.g. charts and graphs. The software is based on a scalable point-and-click user interface, making it accessible for novice users. This software inspired to utilize multiple views in the same window whilst using graphical elements to represent data in a simple and efficient way.
3.3.2 NVivo

NVivo is a qualitative data analysis software that has been designed for qualitative researches that work with rich text-based information, e.g. academic and health researchers. The software makes analysis on small or large volumes of data possible for users (NVivo, 2018). The NVivo user interface is simple and follows common software conventions such as ribbons with symbols and text, an overview window, detailed view, navigation and search bars, Figure 3.2. This software inspired the usage of symbols and short lines of text to represent essential information. The early prototypes were also inspired by the usage of simple colours to clarify selections within this software as well as how this software used tabs as navigation element.
### 3.3.3 MAXQDA

MAXQDA is a software designed to handle qualitative and quantitative research (MAXQDA, 2018). Main functionalities include import, organize, codification, analyse, visualize and publish diverse data sets. The user interface is simple to understand and use but contains a vast number of tools and functions, Figure 3.3 (MAXQDA, 2018). This software inspired to the usage of clear distribution of information as well as how graphical elements can be used to clarify data.

![Figure 3.3 – An example of the user interface of MAXQDA with user customized windows.](image)

### 3.3.4 Dedoose

Dedoose is a web application allowing the user to integrate and combine qualitative and quantitative data analysis methods with interactive data visualizations (dedoose, 2018). The software works on multiple platforms with all kinds of different data sets: text, photos, audio, spreadsheet data and more, Figure 3.4. Dedoose allows codification of qualitative data based on user preferences, integration of quantitative data and interactive mixed methods data visualizations (dedoose, 2018). This software inspired the early prototypes to be designed like web applications – e.g. using familiar patterns and symbols, providing guidance, assistance and feedback as well as making the prototypes intuitive.

![Figure 3.4 – An example on a Dedoose project](image)
3.3.5 Adaptive Insights

The cloud-based Adaptive Insights platform is a performance management software designed for active planning in agile businesses. The software provides a variety of tools and functions making financial planning easy, powerful, fast and accessible from anywhere in the world (Insights, 2018). Adaptive Insights enables processes to become more collaborative, comprehensive and continuous. The software has an in-browser user interface, much like a spreadsheet application with drag-and-drop characteristics, Figure 3.5. Additionally, Adaptive Insights automates integration of data from other systems and platforms which enables collaboration and real-time updates. This inspired early prototypes to use functionalities such as automation of background processes and a smarter system that assists the user whilst collaborating with other users.

![Figure 3.5 – An example of a spreadsheet-like budgeting window in Adaptive Insights.](image)

3.3.6 Adaptive Vision

Adaptive Vision Studio is a machine vision software based on data-flow. The software supports machine vision engineers in inspection and general computer vision. Adaptive Vision supports drag-and-drop functionalities, contains ready-to-use filters and the user can build user filters. An example of this can be seen in Figure 3.6. The software is task-oriented and provides ready-made tools, error handlers and interoperability with C++ and .NET. These functionalities make the software an intuitive, powerful and adaptable product (Future-Processing, 2018). This software inspired to design early prototypes that were fundamentally process oriented meaning that the user can use the same

![Figure 3.6 – An example of how different tools are used to analyse objects in Adaptive Vision.](image)
procedures to obtain different results or to analyse underlying causes. Additionally, the early prototypes were inspired by how this software utilized large icons to represent functions as well as how different views were isolated and made distinguishable.

3.4 Early Prototyping

Based on the persona and the benchmarking study, early prototypes were created. Creating low-fidelity prototypes early is fundamental for usability assurance and a way to facilitate communication (Sikorski, 2014; Weichbroth & Sikorski, 2015). The prototypes that were created were means of representing contexts of use, to inspire and communicate ideas in the design process, not to show any means of high-fidelity functionalities or interaction. The low-fidelity prototypes were made frequently and rapidly using unruled paper and pencils. Examples of the early prototypes can be seen in Figure 3.7.

Figure 3.7 – Examples of early prototypes produced to facilitate communication
3.5 Workshop

One key activity in the early design process is to get users to describe and evaluate upon an existing design, to better understand how the design is interpreted and understood (Costa, Holderb, & MacKinnonc, 2017). Additionally, integrating these types of design practices is necessary to identify causes of design problems and when they arise (Faily, Lyle, Fléchais, & Simpson, 2015).

3.5.1 Procedure

To better understand the problems with MGSS as well as to start of the design process, a workshop was conducted with 13 people who work with the development of the MGSS system. All the attendants had seen and used MGSS before, but not in the same, intended way as the end-users. The main goals with the workshop was to identify pain points within the GUI, to accumulate knowledge on why these pain points existed and how they could be avoided in the future design. Additionally, one goal was to gather thoughts and ideas on how the GUI could be improved.

The workshop was held in a conference room at Saab AB, in an environment familiar to all the employees. The workshop lasted for one hour. The workshop consisted of a short introduction presenting the thesis work, the agenda as well as the goals of the workshop. Then, the group was divided into three groups consisting of four to five individuals each and got the first assignment of the day: Sink the Boat. In this assignment, each group was given a metaphorical picture containing a boat and an anchor as well as several screen-shots of MGSS’s existing GUI. The task was for the groups to identify problems dragging the boat – a metaphor for MGSS – down or holding it back in terms of efficiency, use, flexibility and learnability. The groups were asked to write these problems down on post-it-notes and attach them to the anchor in the picture. During this time, the participants were asked to discuss amongst each other and to encourage – rather than discourage – each other to identify as many problems as possible. Each group was given roughly ten minutes to complete this task, followed by a five-minute presentation where each group presented and explained the problems found. When all the groups had completed this task, the group split up again to complete the second task: Raise the Anchor. In this task, the groups were asked to generate ideas on how the identified problems in the first task could be solved. This was done by writing a brief solution and adding that solution to the respective problem post-it-note. These solutions were then placed above the surface in the picture, making the boat go faster – metaphorically representing an improvement possibility. Each group was given roughly ten minutes to complete this task as well, and subsequently a five-minute presentation to present and explain the solutions.

When the two main tasks of the workshops were completed, a discussion was held with were all groups participated. Lastly, the workshop was summarized and the participants were given feedback on the result and how this result would be used in the coming design process.

3.5.2 Workshop Result

All the groups stated that the appearance of the graphical user interface in general is outdated and needs a clean-up and modernization. One group suggested this could be done by using a gentler colour scheme and a simpler GUI. Another group suggested MGSS follow Saab AB’s graphic profile.

The user interface is not adaptable to user needs. All three groups stated that different users of MGSS use completely different tools and functions in the daily work, and the GUI should be designed accordingly. MGSS always shows every function and tools that the user might possible need, which is inefficient. This could be solved by implementing a customizable user interface with personal user settings and tools that can be saved. One group stated that is also should be possible to hide and show functions when needed and to display functions in a consistent way, to further improve the adaptability and usability of the user interface.
MGSS is a process oriented software in that the user uses the same procedures to obtain different results or to analyse underlying causes, but it is not designed to be process oriented. One example of this is, as one of the groups stated, that operations block and/or lock other operations from operating simultaneously. Another group said that MGSS needs to present the actions needed to perform a specific analysis, instead of showing all functions and tools that may or may not help you to perform an analysis. Today, too many processes within MGSS requires a user input. One group suggested that this makes MGSS inefficient in usage and proposed a solution where processes are automated to decrease the number of steps required to perform analytical tasks. To become more efficient and to increase usability MGSS needs to be designed as a process oriented GUI with the users’ desired way of working in mind.

Two groups stated that MGSS does not help the user in performing tasks and does not assist the user in interpreting data. The user assistance is minimal within the user interface. The same group purposed that the use of a customizable action list could be used to help the user in managing expectations and assisting the users’ daily workflow. Another group suggested that it would be easier to use MGSS if all the information documents and analysis guides that are necessary to perform analytical tasks would be integrated as tools or functions in MGSS. Furthermore, another group stated that the user interface lacks shortcuts that could help the experienced user.

According to the all groups of the workshop, MGSS uses filters inefficiently and inconsistently, e.g. data is filtered and categorized by different classes in different tools, even if the meaning essentially is the same. Filter functions could be used more efficiently by – as one of the groups stated – filtering data grids by specific aircraft or cycles. Additionally, one group stated that filters should be created by the user based to the specific users’ needs.

All groups stated that the Cycle Overview (DCO) window simply is “wrong, ugly and misleading”. The reason for this is that MGSS always shows all data collected. This was a problem identified by all groups; the user interface needs to show less data and only the relevant data in a smart way. One solution to this problem – generated by one of the groups – was to use a single window system with disposable tabs where data can be organized by type. This group meant that many windows leads to time consumption in the users’ workflow. Another solution produced by another group was to use a dashboard-like window as the starting window where the user can choose between a basic and an advanced view.

One group stated that the Download Package Repository (DPR) window contains redundant information. Additionally, the workflow within the DPR is complicated and it is hard to anticipate actions. Another group meant that this applies to the Cycle Information (DCI) window as well; this window presents clickable tabs which does not contain any data.

The Manual Fault Isolation (MFI) window is one of the main windows used by the end-users to troubleshoot a specific cycle, but all the groups identified problems within this tool. One group stated that the MFI does not help the user to troubleshoot based on fault events, which would be helpful. Another group meant that the MFI window lacks consistency in the way it formats variable data. The final group stated that “There is a need of a new MFI function, designed from scratch”.

22
4 Design Process

The design process used in this thesis, Figure 4.1, is an adaptation on the evolutionary software development process (Gilb, 1988). This method reduces risks and costs by examining and analyzing risks in the early phases of a development process (Kesseler & Knapen, 2006). Additionally, it improves user response to change by continuous communication and collaboration which subsequently leads to better product suitable based on users’ needs (May & Zimmer, 1996).

![Graphical representation of the design process used to develop the prototype.](image)

To successfully meet the needs and goals described in the persona a participatory design approach was used in the design process. Based on the fundamental aspects of participatory design formulated by Halskov and Brodersen (2015) two focus groups were constructed; the primary and the secondary focus group. The members of the primary focus group were chosen based on the Politics and People aspects of participatory design. The participants in the primary focus group were the responsibility domain manager (MGA) of MGSS and three MGSS software developers. The primary focus group participated in important design decisions and prototype evaluation throughout the design process. The secondary focus “group” was composed by the flight test engineer who also participated in the interview mentioned earlier. The flight test engineer was consulted to test and evaluate the prototypes in the proof-of-product and proof-of-value prototyping session. None of the participants in the design process had any prior design experience.

4.1 Proof-of-context prototyping

In this section the design of ten proof-of-context prototypes will be described in brief, followed by a description of the testing session that was conducted and subsequently, the results from the focus group’s evaluation.

4.1.1 Design

Based on the workshop results and the early prototyping sessions, ten static, low-fidelity paper prototypes (Arvola, 2014) were created to explore and understand design ideas regarding context of use, related to user requirements presented in the persona. Furthermore, the paper prototypes were created to decide what design approach to consider in following proof-of-concept prototyping session.
The prototypes were quickly designed on A4-sized paper, using minimum amounts of colour and post-it-notes. Colours and post-it-notes were only used to emphasize certain important functionalities or to further explain the design, if needed. The ten prototypes were designed to represent a potential main window or dashboard within the new GUI and none of them were interactive.

Same, Same, but Different
This idea was meant to visually look like MGSS today, but were meant to better assist the user in performing analytical tasks by helping the user to interpret fault events. Hence, the name "Same, Same, but Different". Additionally, this prototype was meant to be more customizable, based on user preference, and have a simpler main window with a simple workflow. This prototype contained different import functions, two major data set windows; one overview window and one where the program has calculated and organized fault events by criticality, four tool buttons that open tools based on selection and an export function, Figure 4.2.

Command System
The second prototype was inspired by Microsoft Visual Studio IDE (Microsoft, Visual Studio, 2018) and the NVivo software. This prototype allowed the user to create and customize desired processes and tools, and was named "Command System". This could be done by typing commands in the command bar. This prototype was made for more experienced users that already know what functionalities the system contains. Instead of opening new windows for every function or tool, this prototype was a tab-based user interface. The prototype contained an import button, a profile view where completed commands appear, a command view, a command bar, a tools view and a data set window with tabs for data representation Figure 4.3.

Module Based
This prototype was inspired by the SAS/STAT software and was named "Module Based". The idea was to make a simple GUI that helps the user in evaluating and classifying data sets and subsequently assisting the user in choosing necessary tools to be used in analysis. This prototype was supposed to be more assessable to novice users. This prototype contained different import buttons, a matrix of functions that will unlock when data is imported and two summary views, Figure 4.4. After the system has classified specific data sets, it places them in the modules to be further analysed and the user is then allowed to open a new window containing the specific tool with the respective functionalities.

Adaptive Software
The fourth prototype was inspired by Adaptive Vision, described in section 3.3.6, and has adopted functionalities such as drag-and-drop, customizable processes and adaptable to multiple users and purposes. This prototype was named "Adaptive Software". The idea was to make a prototype that assists multiple users whilst making the daily work of any user more efficient by using user specific processes. This prototype contained a toolbox view containing all functions and tools that exists in MGSS today, a properties view, a process editor where functions and tools could be dragged-and-dropped from the toolbox view and several, customizable data set representation views, Figure 4.5.

Contextual Application
The fifth prototype was substantially different from the existing MGSS GUI. This prototype was inspired by dashboard software packages often used in statistical or financial management, such as Dedoose. This prototype was named "Contextual Application". This prototype was meant to be entirely customizable, based on settings in a user profile to increase flexibility and efficiency. The prototype – in this session – contained five different overview functions, four data set representations and one output function Figure 4.6. All functions and tools with the respective functionalities opened in a new window when clicked.
Split Vision
This prototype was created to improve overview without opening new windows to represent data, and was named “Split Vision”. This prototype was inspired by MAXQDA and Blender (Blender, 2018) - an open source 3D creation suite. This prototype contained import functions, a functions view and a customizable main data view in which tools with data sets can be presented, Figure 4.7. Tools appear in the main data view by drag-and-drop from the functions view. The tools in the main data view can be maximized and minimized within the main window to facilitate comparability and overview.

Simply Advanced
The seventh prototype was partly inspired by insights from the workshop and the persona, and partly by the Adaptive Insight software. In this prototype, named “Simply Advanced”, the user is assisted in performing tasks by categorization of imported data, evaluation of data, shortcuts, graphical workflow assistance and simple output functionalities, Figure 4.8. The main window contained an import button, an imported data view, symbols to indicate the next step in the analysis and to represent criticality, different export functions, several tool buttons and two data set representation views.

Advanced + Simple
The eighth prototype was based on the prototype designed in the related work of this thesis (Magnusson, 2016). This prototype is a graphical representation of all the users’ needs identified by Magnusson (2016). It was based on the idea of a GUI were the user can choose to use a simple or advanced view, in which analysis can be performed. The prototype created was named “Advanced + Simple” and in this session the simple view is presented, containing an organized data view, a simplified functions view with the respective data grid, a cycle history view where exported data sets appear and a search view where the user can easily search the AMP, Figure 4.9. In this session the advanced view was not prototyped.

Dashboard
Inspired by the Windows 10 dashboard (Microsoft, 2018), the ninth prototype was named “Dashboard”. This prototype contained a graphical and customizable main window in the form of a dashboard, Figure 4.10. The prototype contained functions and tools in the form of widgets added by the system by default, based on usage frequency. Additionally, the users can add widgets to use in the analysis process. A new window opens from the dashboard when the user opens widgets and several widgets can be selected to perform a specific analysis.

Export Focus
The last prototype was created by insights from the workshop. MGSS is mainly used to export data with several different internal and external tools, therefore a prototype named “Export Focus” was created to mainly support export functionalities. This prototype was created with export as the user’s main goal and analysis as a sub-goal. It contained a customizable dashboard view where export oriented tools were presented together with overview tools, Figure 4.11. Additionally, the main window presented an analysis view where further analysis can be conducted if needed. This prototype included three post-it-notes. Out of these three notes, two presented all the internal and external exports that can be performed in MGSS today to emphasize the amount of export options available, and one note to represent a new version of the “Cycle Statistics” window.
Figure 4.2 – Same, Same, but Different

Figure 4.3 – Command System

Figure 4.4 – Module Based

Figure 4.5 – Adaptive Software

Figure 4.6 – Contextual Application
4 Design Process

Figure 4.7 – Split Vision

Figure 4.8 – Simply Advanced

Figure 4.9 – Advanced + Simple

Figure 4.10 – Dashboard

Figure 4.11 – Export Focus
4.1.2 Testing

The testing of the proof-of-context prototypes was conducted for one hour at a conference room at Saab AB. Everyone included in the primary focus group participated in the testing. The main goal with the testing session was to decide on which design approach to take when designing new concepts, based on usability criteria (Shackel, 1986).

The ten prototypes were presented one at the time to the participants, who all sat by a small table. Each prototype was presented for roughly 5 minutes. The focus was to explain where the inspiration came from, why the prototypes were designed in a certain way, what the outcome of this design could be and how the final product was meant to work. After the individual presentation, all prototypes were placed on a white board. The participants in the primary focus group were then asked to individually rank the three best paper prototypes using coloured white board magnets. The participants were asked to consider the following criteria; efficiency improvement; enhanced usage; increased flexibility and improved learnability. The participants could place all the respective three votes on the same paper prototype. The primary focus group participants' votes are presented in Table 4.1.

<table>
<thead>
<tr>
<th>Prototype Description</th>
<th>Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dashboard &amp; Export Focus (these two prototypes were essentially the same in the eyes of the focus group participants and were therefore merged together)</td>
<td>6 votes</td>
</tr>
<tr>
<td>Advanced + Simple</td>
<td>2 votes</td>
</tr>
<tr>
<td>Same, Same, but different</td>
<td>2 votes</td>
</tr>
<tr>
<td>Adaptive Software</td>
<td>1 vote</td>
</tr>
<tr>
<td>Contextual Application</td>
<td>1 vote</td>
</tr>
</tbody>
</table>

4.1.3 Evaluation

After the test, a participative evaluation session was conducted where the primary focus group were asked to explain the respective ranking and why the choices were made in a certain way. Additionally, the participants were encouraged to suggest improvements to be implemented in the following proof-of-concept prototyping session.

The primary focus group stated that a perspicuous prototype approach was needed, the user does not want to see as much information and data as presented in today’s MGSS. “Dashboard” and “Export Focus” prototypes were the most promising in response to the given criteria, according to the unanimous primary focus group. One of the participants stated that the “Dashboard” approach would be the easiest way to make novice users understand and use MGSS. Another participant agreed and added that the “Dashboard” prototype would be best suited for many different types of usage thanks to the apparent flexible approach.

Additionally, the primary focus group stated that essential information should be presented in an efficient way. One efficient way of doing that was like the colour categorization in the “Contextual Application” and “Same, Same, but Different” prototypes. Two of the participants in the primary focus group appreciated the way essential information was presented in the “Same, Same, but Different” prototype. Two other participants appreciated the idea of a simple and an advanced view, like the one presented in the “Advanced + Simple” prototype. All participants agreed on the usage of tabs, e.g. Command System, being superior to the usage of windows. One participant stated that the future
MGSS needs to be process oriented to facilitate efficient usage, and suggested this could be done with the “Adaptive Software” approach.

Design conclusion
According to the primary focus group, the Dashboard and Export Focus context prototypes were the most preferable in terms of efficiency improvement, enhanced usage, increased flexibility and improved learnability. Still, promising features and ideas from the other prototypes could not be excluded at this early stage. If the other prototypes with their respective features were to be excluded, the fundamental mean of usability would partly be lost. All the ten proof-of-context prototypes could potentially improve the usability of MGSS as they were representations of user needs and requirements. For instance, the Adaptive Software prototype represents a process oriented and customizable user interface that gives every user control over their respective workflow which in turn increases the efficiency and learnability; and the Command system context prototype provides a tab-based user interface for experienced users, that supports flexibility and subsequently increases efficiency. However, the Dashboard as well as the Export Focus prototype had the most modern approach. Additionally, these prototypes had the potential to increase flexibility and efficiency as well as being use-enhancing through a customizable, familiar, flexible and simple design. Furthermore, these prototypes corresponded most accurately to the scenario described in the persona. Therefore, the “Dashboard” and “Export Focus” prototypes were concluded to be the foundation of the contextual approach in which concept prototypes were to be further developed.

4.2 Proof-of-concept prototyping

In this section the design of the five proof-of-concept prototypes will be described in brief, followed by a description of the testing session that was conducted and subsequently the results from the primary focus group’s evaluation.

4.2.1 Implementation

Based on the design conclusion of the first prototyping session, five ideas were implemented and designed in the proof-of-concept prototypes: an entirely customizable Dashboard concept; a combination of Dashboard and features of Adaptive Software; a combination of Dashboard and features of Command System; a combination of Dashboard and features of Same, same, but Different; and a combination of Dashboard and features of Advanced + Simple.

4.2.2 Design

Five interactive, low-fidelity paper prototypes (Arvola, 2014) were created to investigate the usability of the prototype concepts and furthermore to decide what concept design to implement in the design of the product. Additionally, the prototypes were created to further explore design functionalities to meet user requirements. The prototypes were quickly designed by using A3, A4 papers and post-it-notes. This was done to resemble the correct format of the final product. The A3-sized papers were used to resemble static tools and functions, as well as the starting view of all prototypes. Whereas the smaller papers and post-it-notes were used to emphasize important functionalities and interactive elements. The five prototypes were designed to represent a sequence of the most common actions needed in the daily work of the user.

General elements

Several general elements were created and used in all the five prototypes. These elements were: a representation of the DCI, Import and Export window, an element representing the AMP, an information note and an error note, Figure 4.12.
Clean Setup
The first concept was named Clean Setup and was based on the idea of an entirely customizable dashboard, where the user can choose what functionalities to use within MGSS. In this concept, the user is first met by an empty workspace apart from two interactive elements: an interactive start button and a help button, Figure 4.13. When pressing the start button, a tools window opens where the user gets presented with a list containing tool icons. In this list, all tools and function that exist within MGSS can be found. The icons have an information box, where information about the usage of the specific tool is presented if clicked. Additionally, each icon has a check box that lets the user choose what tools to import into the dashboard of MGSS. The user imports the chosen tools by using the insert button. The chosen tools then appear as icons on the initial workspace together with an icon containing a plus sign. This element can be used if the user wants to add more tools to the workspace, and will present the tools window if clicked. If the user wants to remove tools from the workspace, a close button that is placed on every tool icon can be used. To use different tools in the workspace, the user double-clicks the icon and a new window opens. This new window contains tool specific information and functionalities, like the ones presented by MGSS today, e.g. DCI and Export. Every tools’ window assists the user in performing task and gives feedback when a task is completed.

Adaptive Dashboard
The second concept implemented the process oriented features of Adaptive Software to the Dashboard approach, and was named Adaptive Dashboard. In this concept, the user is presented with a workspace containing several elements from the beginning: a process view containing a grid and a start process button; numerous tool icons with a close and an information button; and the same plus sign icon as presented earlier, Figure 4.14. The user is met by an empty process view when using the system for the first time. To create a process, the user can drag-and-drop tool icons into the process grid. The tools transform into text boxes containing the name of the tool, arrow buttons and a close button that can be used arrange tools into a desired order of implementation, and to remove a tool from the process. When the user has created a desired process, the process can be started by clicking the start process button. The user is then presented with a new window containing the first tool in the list. This new window contains tool specific information and functionalities, like the tools presented earlier. When the user has completed desired tasks within that tool, the next tool window in the list opens and this procedure continues until every tool from the list has been used. When a process is completed, the user gets feedback and is then presented with the initial workspace where the last used process remains in the process view. Every new tool window assists the user in performing tasks within that specific tool and gives feedback when a task is completed.
Tab-Board
The third concept was inspired by the fact that the primary focus group stated that the usage of tabs is superior to the usage of windows, hence the name Tab-board. In this concept, the initial workspace contains several tool elements, the same plus sign element as presented earlier and an empty tab bar, Figure 4.15. When tool icons are interacted with, the tab bar is filled with the respective tools. When a specific tab is clicked, the corresponding tool and the tab bar is presented in the workspace. The tabs can be removed by clicking the close button on each tab. Then user can change what tool to view by interacting with the tabs. Additionally, the user can choose to view the tools in new windows instead of tabs by dragging the tools from the tab bar and dropping it in the workspace. If desired, the user can return to the initial workspace at any time by using the return button. The tool tabs – and tool windows – contain tool specific information and functionalities, like the tools presented earlier.

Same, Same, but Dashboard
The fourth concept implemented the presentation and categorization features of the Same, Same, but Different prototype to the Dashboard prototype, and was named Same, Same, but Dashboard. In this concept, the user is presented with an initial workspace containing tool icons and three different list views: Acceptable, Analyse and Completed, Figure 4.16. Tool icons can be removed and added similarly to the other concepts. When cycles are imported into the workspace, the system classifies all cycles based on the respective fault code type. The cycles are then organized and placed in either the Acceptable list, if the system can interpret the data without finding any uncommon fault codes, or in the Analyse list, if the system finds uncommon fault codes that need to be further analysed by the user. The user can then select cycles individually and choose what tool to use to interpret the fault codes. This is done by selecting a specific cycle in e.g. the Analyse list and then by clicking specific tool icons in the workspace. Multiple cycles subsequently opened in multiple tools simultaneously. A new window containing tool specific information and functionalities, related to the specific cycle is then opened. When a cycle has been analysed, checked and saved, the system places that cycle in the Acceptable List. From there, the user can select to export individual or multiple cycles. When cycles have been exported from the Acceptable List, this concept helps the user keep track of exported cycles by listing all in the Completed list.

Activity Dashboard
The fifth and final concept prototype implemented the simple and advanced view feature of the Advanced + Simple to the Dashboard approach, and was named Activity Dashboard. This concept is different from the previously presented in that the first window that is presented to the user is an import window. When cycles have been imported, the main workspace is presented. This concept is like the Same, Same, but Dashboard concept in that the initial workspace contains lists but instead of result oriented lists, e.g. the Analyse list, the lists are process oriented. Additionally, this concept has two different modes: a simple and an advanced mode, Figure 4.17. In the simple mode the user gets an overview of analysis that has been done previously, impediments that have occurred and cycles that yet have not been analysed. This overview is presented in three lists: The Yesterday, Impediment and Today list. Individual or multiple cycles as well as the whole list can be selected to be used in the advanced mode. If a specific cycle or an entire list is selected, the user can go into advanced mode. This allows the user to bring that specific data into the advanced mode. There, the user can perform analysis on cycles from the different lists using various tools, similarly to previous concepts. When cycles have been exported from the Today List, this concept helps the user keep track of exported cycles by listing all in the Completed list. The cycles in this list would be transferred to the Yesterday list the following day, if there were no impediments.
Figure 4.13 – Examples of interactive elements used in the Clean Setup concept

Figure 4.14 – Examples of interactive elements used in the Adaptive Dashboard concept

Figure 4.15 – Examples of interactive elements used in the Tab-board concept
Figure 4.16 – Examples of interactive elements used in the Same, Same, but Dashboard concept

Figure 4.17 – Examples of interactive elements used in the Activity Dashboard concept
4.2.3 Testing
Prior to the testing of the interactive proof-of-concepts paper prototypes, a pilot test was conducted with two novices at Saab AB. The novices had no previous knowledge on MGSS, nor had they seen or tested the system before. This test was carried out to detect faults in the test and additionally to test the understandability of the different concepts as well as the created use case. Additionally, this testing session gave an indication of expected performance time of each task. This information was used to structure the testing session. This pilot test was conducted three days prior to the testing session of the concept prototypes.

The testing of the proof-of-concept prototypes was conducted as an assisted evaluation process. The session was held for one and a half hours, at a conference room at Saab AB. Everyone included in the primary focus group participated in the testing. The testing session was conducted to observe user interaction with the developed concept prototypes whilst testing learnability and efficiency when performing a predefined use case. This testing session was conducted twelve days after the proof-of-context testing session.

One use case was created and the participants were requested to, individually, try and perform actions by interacting with the different paper prototypes. The use case was based on the users’ main goals, described in the persona. The use case that was presented to the participants was:

1. Import the cycle named “394.86-2018-04-10 15:50:02”.
2. Analyse the Fault Report in the Cycle Information tool (DCI).
3. Suggest a solution to the fault presented, then check and save the Fault Report.
4. Perform an export on the given cycle.

Based on the participant’s interaction, e.g. selections and clicks, the prototypes altered by introducing new elements, dependent on the systems intended reaction. Only the intended reactions of the paper prototype had been designed and if the participants were to interact in an unintended way, the note with the text “Error!” was presented to the participant.

All participants were asked to perform the same use case with all the five different paper prototypes. When one participant performed the use cases, the others were asked to leave the conference room to not be influenced by each other’s ways of solving the use cases as well as ways of interacting with the prototypes. The participant sat at a table opposite the facilitator and was not able to see interactive elements, e.g. new windows or post-it-notes, that was not in use. This prevented the participant from guessing what element to interact with based on what elements were about to be presented. Each participant was given twelve minutes each to perform the presented tasks. While performing the use case, the participants were asked to think out load and explain the respective actions. The facilitator only assisted the participant if difficult problems occurred during the test. The amount of interactions resulting in the “Error!” element was noted to determine understandability and subsequently, learnability. When all participants had performed the use case with the five different prototypes, the participants were asked to individually rank the three best paper prototypes using coloured white board magnets. The participants were asked to consider the following criteria; efficiency improvement and improved learnability. The participants could place all the respective three votes on the same paper prototype.

4.2.4 Evaluation
After the testing session, the proof-of-concept prototypes were discussed and evaluated based on the participant’s use case performance and respective ranking result. Additionally, the participants were encouraged to suggest improvements to implement in the following proof-of-product prototyping session.
The primary focus group participants stated that the Clean Setup concept required too many user interactions and furthermore, it was stated that the concept was not perspicuously process oriented. This led to the Clean Setup concept being the least efficient and hardest to learn, according to the participants of the primary focus group. This is reflected by the average amount of errors as well as the number of votes in Table 4.2.

As opposed to the Clean Setup concept, the Same, Same, but Dashboard concept reflected a definitive and perspicuous process orientation that better assists the user, according to the participants of the primary focus group. All participants agreed on the concept being the most efficient concept as well as the most understandable. This is reflected by the average amount of errors, presented in Table 4.2.

The participants stated that the new MGSS needs to show what previously has been done and what needs to be done without presenting unnecessary information to the user. One effective way of doing this, was emphasized by the participants: “using lists is the best way to organize enough information on which the user can base decisions”. One suggestion from one of the participants was to implement the lists from the Activity Dashboard concept into the Same, Same, but Dashboard concept. Furthermore, this concept gave the user the ability to customize the dashboard based on user needs and requirements. This was a desired feature according to two of the primary focus group participants. Two other participants instead suggested that one default dashboard layout should be implemented when the MGSS starts.

Table 4.2 – Amount of errors and distribution of participants’ votes per concept

<table>
<thead>
<tr>
<th>Concept</th>
<th>Average amount of errors [n]</th>
<th>Number of votes [n]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean Setup</td>
<td>2.3</td>
<td>0</td>
</tr>
<tr>
<td>Adaptive Dashboard</td>
<td>2.3</td>
<td>0</td>
</tr>
<tr>
<td>Tab-board</td>
<td>0.5</td>
<td>2</td>
</tr>
<tr>
<td>Same, Same, but Dashboard</td>
<td>0.5</td>
<td>8</td>
</tr>
<tr>
<td>Activity Dashboard</td>
<td>0.8</td>
<td>2</td>
</tr>
</tbody>
</table>

Three participants agreed on the fact that usage of tabs, similarly to the Tab-board concept, was an effective way of displaying and using tools as well as functions. Additionally, the primary focus group stated that the usage of tabs was a good way of making elements in a process easily accessible. Two of the participants suggested that the Same, Same, But Dashboard main window should be the first default tab in the new MGSS. Another participant suggested that a notification view that presents process progression as well as completed work would enhance the usage and increase the understandability of a tab-based dashboard.

The Adaptive Dashboard concept was problematic and not a viable concept, according to the participants of the primary focus group. It was stated that the processes only occasionally are performed in the way presented in the prototype; performing all tasks on one cycle before analysing the next cycle. Additionally, two participants stated that this type of process orientation made the
concept less adaptive and more confusing, as well as inefficient. Instead, the participants stated that the user wants to perform one task, e.g. export, on all analysed or confirmed cycles simultaneously.

The primary focus group participants emphasized the need to only see basic information and status of specific cycles with the option to expand windows to get more information. One participant suggested that the basic information that needs to be shown is cycle status, type and an icon.

**Design conclusion**
According to the primary focus group, the *Same, Same, but Dashboard* prototype was, by far, the most preferable in terms of usability. This prototype presented interactive elements and data in a perspicuous and simple whilst representing a familiar environment. Additionally, functions were represented by easy-to-use buttons and making a fault analysis in this prototype seemed to be relatively simple. These factors correspond to desired users’ requirements as well as needs. However, the prototype lacked the desired process orientated GUI to support different users’ way of working. Furthermore, the prototype was not flexible enough to support user’s ability to modify specific actions and processes as well as being able to perform actions in several different ways. The *Tab-board* prototype had a more flexible design approach but lacked perspicuous elements such as a simple overview window. The *Activity Dashboard* lacked flexibility but was process oriented through a clear and understandable “start-to-finish” work flow. Therefore, the *Same, Same, but Dashboard* prototype was concluded to be the concept foundation on which a proof-of-product prototype were to be further developed and more thoroughly designed.

4.3 Proof-of-product prototyping

*In this section the design of the proof-of-product prototype will be described in brief, followed by a description of the multiple testing session that was conducted and subsequently the results of the focus groups’ evaluation.*

4.3.1 Implementation

Based on the design conclusion from the previous prototyping session, improvements to be implemented were represented by means of co-developing a new concept prototype with the primary focus group participants. In this concept prototype, favourable features and aspects of the above-mentioned proof-of-concept prototypes were included, Figure 4.18. The new combined and enhanced concept prototype included elements of the *Same, Same, but Dashboard* and *Tab-board* concept combined with various other concept features, e.g. lists from the *Activity Dashboard* concept.

![Figure 4.18 – Representation of the fundamental aspects of the co-created concept](image)
4.3.2 Design

One interactive and high-fidelity prototype was created based on the co-developed concept. This prototype was created by means of investigating the usability of the new GUI product. The prototype was designed using Adobe XD, a user experience design software application. This prototype was designed to better resemble the final user interface prototype in terms of interactivity, basic visuals, content and navigation hierarchy. The colours used were based on the graphic charter containing rules regarding the graphic identity of Saab AB. The resolution 1920x1200 pixels was used to design the framework of the GUI. This was done to suit the computer monitors used by the end-users. The prototype was created to represent a sequence of common tasks and actions needed in the daily work of the end-users. The following figures describe one of the common tasks used by the end-users – importing cycles, analysing cycles using the Fault Report tools and then exporting cycles.

The interactive prototype can be seen and tested on this link: https://adobe.ly/2I7bMmW

The starting view of the proof-of-product was designed to better assist and guide the user in performing actions, hence the emphasized import function as seen in Figure 4.19. Additionally, this design decision simplified the GUI, making it less overwhelming to the user. The starting view contained numerous elements, e.g. lists and function buttons, with low opacity. These elements were designed to give the user an indication of elements to be used later in the user’s work flow, allowing action anticipation and increased guidance. Different elements within the GUI were placed in a way to provide a familiar and linear, left to right oriented process work flow. When the user interacts with the import button, a tab with a folder explorer will open where desired cycles can be selected and imported with fewer actions compared to the existing MGSS. To not cause frustration and to decrease confusion, the user is presented with a progress bar while the cycles are being imported.

Figure 4.19 – The starting view of the proof-of-product prototype
When the chosen cycles have been imported, the GUI assists the user in classifying and filtering the cycles based on fault events and fault codes – placing cycles with familiar, non-critical fault events in the *acceptable* list, cycles with critical, unfamiliar fault events in the *analyse* list, the recently imported cycles in the *today* list and already evaluated and exported cycles in the *completed* list. Additionally, the GUI notifies the user that an action has been completed by placing a notification icon on the upper, right-hand side of the GUI, as seen in Figure 4.20. This notification panel can be expanded to give the user information on occurred events and performed actions. The GUI presents several lists containing numerous interactive elements. Every element in the list represents a cycle with an icon, indicating cycle type. Additionally, two lines of text are implemented by means of giving the user additional information about the specific cycle, e.g. cycle number, cycle time and date. This information is supposed to give the user a more perspicuous overview, where only essential information is presented. However, if the user needs more specific information expandable windows can be obtained by interacting with the small arrows placed next to each cycle, as shown in Figure 4.20. Objects in the lists can be selected individually or collectively to perform further actions within the GUI. When a selection has been made the GUI assists the user by presenting suitable actions and/or tools to perform with the specific cycles whilst hiding less suitable tools, i.e. if all cycles in the *analyse* list are selected the GUI presents several analytical functions whilst hiding the export function. This functionality assists the user in performing the correct tasks, decreasing faults and errors related to usage. When cycles in the analyse window have been selected, the user can choose any analytical tool to analyse or troubleshoot the cycles, e.g. *Fault Report* or *MFI*. When interacting with a tool, tabs containing cycle specific information on fault events and fault codes will appear.

Figure 4.20 – The main view of the prototype when cycles have been imported.

When, in this case four, cycles have been selected to be analysed using the *Fault Report* tool, four tabs will open as seen in Figure 4.21. One tab corresponds to one cycle with its specific fault events and fault codes. The user can interact with the different tabs to change what fault report is viewed as well as drag them from the tab bar to view multiple fault reports simultaneously. From this singular fault report view, the user gets a simple overview of the most important information used to troubleshoot fault events; occurred fault reports appear in the upper left rectangle, the corresponding fault codes and variables appears in the upper right rectangle and in the bottom left the existing fault code view of today has been extended to assist the user in interpreting and understanding fault reports. In the
bottom left rectangle, the user is presented with suggestions on how to interpret the fault code based on information in the AMP. This has been implemented to simplify the analytical process subsequently making this process more effective. The user is also supposed to easily retrieve essential information from the AMP by clicking the Open AMP button, giving the user control whilst making the assumptions made by the GUI transparent. From this view, the user needs to check and save each fault report individually to be able to classify this cycle as evaluated and therefore exportable, similarly to how this procedure works in MGSS today. The user gets feedback via an information window as well as a note in the notification panel when a specific cycle has been evaluated. An additional information box is presented when all cycles from the analyse list have been checked and saved. When one or several cycles from the analyse list have been saved, the GUI automatically classifies them as acceptable and moves the respective items to the acceptable list in the main view.

![Figure 4.21 – The Fault Report view with four different tabs](image)

When all cycles have been evaluated and checked, they are transferred to the acceptable list in the main view. By selecting items in this list, the user can either see additional information by clicking the small arrow to expand an information window or export the selected cycles via the export tool, as seen and emphasized in Figure 4.22. The information presented in the expandable window could present when the cycle was evaluated, what types of fault events that has been evaluated or information concerning the type of export that needs to be done.
Similarly, the GUI in this view assists the user in performing actions and guides the user towards the next step in the analytical process. In this view, this is done by lowering the opacity of tools not intended for usage with this specific type of cycle, e.g. Fault Report since the cycles already have been evaluated and saved. When one or several items in the acceptable list is selected, the export tool can be used. When the user uses this tool, a new export tab opens from where the selected cycle can be exported as seen in Figure 4.23. In this prototype, all different kinds of exports are gathered in the same tool instead of being distributed over different places, as they are in MGSS today. From this export tab the user can open the folder explorer and choose to what directory the selected cycle should be exported. Additionally, the user can return to the main view by clicking the home tab. This option is always possible to increase the flexibility of the prototype. When the cycle has been exported, the GUI again transfers the cycle to another list – the completed list. From here the user is presented with information on when the export was completed. Furthermore, the user gets feedback that a specific cycle has been exported, and if all cycles are exported simultaneously, the user receives similar feedback.

All actions and tools used in the sequence presented above are presented in the notification panel to the right in Figure 4.23. This information can be achieved by clicking the icon with three vertical lines, which will expand the panel. This functionality is crucial to help the user get control of the situation and respective workflow – the panel presents what has happened as well as presenting when and why something happened. This is also useful to reduce cognitive workload.
4.3.3 Testing

Prior to the testing of the interactive proof-of-product prototype, a pilot test was conducted with a system engineer at Saab AB. This pilot test was conducted three days prior to the product prototype testing session. This test was carried out to detect faults in the prototype design and additionally, to test the understandability of the different design elements. The test was conducted as a walkthrough of the prototype. After conducting the test, the system engineer emphasized the need of filtering tools as well as always being able to reverse actions. Additionally, the starting window had to be changed from a mandatory import window to an overview window where the import function instead was a function. The reason for this was to better resemble the system environment of today’s users as well as to support different user work flows. Furthermore, the need of four different lists and their respective names were questioned. The today list was confusing and essentially served the same purpose as the completed list. The today list was concluded to be removed to only present vital information to the user.

Figure 4.24 – Examples of changes that was implemented after the pilot test.

Figure 4.23 – Example of the export view as well as the notification panel
The testing of the proof-of-product prototype was conducted as a participative evaluation process in a context related environment. This testing session was conducted ten days after the proof-of-concept testing session. The session was held for one hour, at a conference room at Saab AB. The different views of the prototype were presented one at the time on a TV monitor, allowing each participant to clearly see the prototype. The participants were encouraged to ask questions during the presentation. Each view was presented for roughly 5 minutes. Everyone included in the primary focus group participated in the testing. The testing session was conducted to investigate user’s impression on the design, e.g. what the participants thought specific elements did or what the expected results when performing an action was. After the first testing session, a second similar testing session was conducted with the participant of the secondary focus group. The second testing session was conducted to investigate end-user’s impression on the design.

4.3.4 Evaluation

After each of the testing sessions, the proof-of-product prototype was discussed and evaluated based on the following criteria; efficiency improvement; enhanced usage; increased flexibility and improved learnability. The participants of both focus groups were encouraged to suggest improvements to implement in the following proof-of-value prototyping session.

The participants of the primary focus group stated that the proof-of-product prototype was easier to understand and use than the existing system, for various reasons; a more logical process oriented workflow than before, easier to find and anticipate object location as well as a simple but flexible design which facilitates navigation. Furthermore, it was stated that the prototype improved the effectiveness and learnability of the system by means of using standardized work flows where the user gets constant feedback on actions. However, the participants stated that certain interactive elements could be further emphasized and de-emphasized respectively to improve guidance and feedback of user actions. Additionally, it was stated that grouping and classification of objects could be used more thoroughly and clearer. One problematic aspect of this prototype design was, according to the focus group, the fact that a dynamic product often cause trouble in knowledge as well as information transfer between users.

The impression of the participant of the secondary focus group was largely positive, but the participant stated that the proof-of-product prototype lacked a few necessary functionalities and tools. A few ideas that would improve the learnability, flexibility and enhance the daily usage were suggested, e.g. implementing a more detailed import window, develop an explorer for the DTU and implement an overview tool with filters. The participant then emphasized the need of a perspicuous and simple user workflow as well as the users’ ability to modify and customize the user interface based on user specific needs. It was stated that the proof-of-product prototype solved the complex problem of interpreting fault events in a simple, smart and effective way. Additionally, the participant stated that the design of the prototype felt modern.

Design conclusion

Even though the overall impression of the design was positive several changes could be implemented to further increase the usability of the prototype. The essential changes are presented in Table 4.3.
Table 4.3 – Concluded changes to be implemented in the final prototype.

<table>
<thead>
<tr>
<th>Usability characteristic</th>
<th>Essential changes to implement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Learnability and use-enhancement</strong></td>
<td>The colour contrast needed to be increased in the Fault Reports window. The contrast was not high enough to distinguish and separate all interactive elements. Additionally, by increasing the contrast in that specific function the colour consistency of the GUI could be increased. To increase learnability, minor language changes, e.g. “check” needed to be changed to “confirm” to better resemble the existing system language.</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td>The AMP button had to be made interactive to demonstrate how this function and its functionalities could be used more effectively within the new GUI.</td>
</tr>
<tr>
<td><strong>Flexibility and use-enhancement</strong></td>
<td>The DCI functionalities of today’s MGSS had to be implemented in the Fault Reports window, making it possible to examine fault events as well as fault reports. Previously, this was not possible but the DCI was a function that had to meet the users’ needs and requirements of the daily usage of MGSS. This had to be done by implementing expandable lists in the Fault Report window. This way, only essential information could be presented but would also allow the user to expand and contract information boxes when necessary.</td>
</tr>
<tr>
<td><strong>Learnability and use-enhancement</strong></td>
<td>To enhance usage and to increase learnability the Add tools function had to be structuralized and simplified. Furthermore, the usage of this function was indistinct and needed to be clarified. A search function within the function, information boxes and a method to support direct single function usage through the Add tools functions had to meet the users’ needs. The name of this function was confusing and needed to be changed to better resemble its functionalities.</td>
</tr>
<tr>
<td><strong>Use-enhancement</strong></td>
<td>A Cycle history function making it possible to get an overview of all imported cycles had to be implemented to meet users’ requirements.</td>
</tr>
<tr>
<td><strong>Learnability and efficiency</strong></td>
<td>The import function needed to be more explicit to better assist the user in anticipating what types of action would be needed to be utilized in the coming work. Enabling different kinds of exports functions within the export was also necessary to decrease the number of actions needed to use of one of the most used functions in MGSS.</td>
</tr>
<tr>
<td><strong>Flexibility and use-enhancement</strong></td>
<td>A function to support customizable lists as well as different standardized dashboards was implemented. This made the GUI more customizable which encourages flexibility.</td>
</tr>
</tbody>
</table>
Use-enhancement and learnability

Examples of text to be used in information windows, functions and overview functions needed to implement to demonstrate how information can be used in the new GUI. Similarly, a DTU-status bar and standard tabs were desirable to better resemble the final product as well as to demonstrate potential usage.

Efficiency and use-enhancement

Even more reversibility had to be implemented. The user always needs to be able to return to the last page or to reverse faulty actions. Additionally, more transparency and feedback needed to be implemented to better assist the user as well as increasing action anticipation.

4.4 Proof-of-value prototyping

In this section the design of final prototype – the proof-of-value prototype will be described, followed by a description of the testing session that was conducted to evaluate the value of the prototype and subsequently, the results from the focus group’s final evaluation.

4.4.1 Implementation

Based on the design conclusion from the previous prototyping session, several elements in the proof-of-product were changed and additional elements were implemented to improve the overall usability of the GUI: the colour contrast of elements as well as additional, familiar icons were needed to be implemented to increase the user’s cognitive ease; more reversibility in terms of return functions had to be implemented to increase the flexibility of the prototype; illustrative text and visual elements as well as standardized functions were needed to be implemented in order for the prototype to better resemble the real product; additional functions, e.g. customizable lists and an extensive export function, had to meet user requirements; and lastly, information throughout the prototype needed to be made more explicit to better resemble desirable functionalities, increasing the understandability of the prototype.

4.4.2 Design - Result

This section introduces the final prototype and explains important design elements in the GUI. The prototype was designed using Adobe XD. This proof-of-value prototype was designed to resemble the final user interface in terms of interactivity, visuals, content and navigation hierarchy. Additionally, the prototype was created by means of inspiring future software development of MGSS in terms of usability. The prototype was created to represent a sequence of common tasks and actions needed in the daily work of the end-users.

The proof-of-value prototype can be seen and tested on the following link:
https://adobe.ly/2FXxGr2
The starting view has a minimalistic design, presenting only essential information and interactive elements to user, Figure 4.25. Additionally, the starting view provides a simple, understandable overview containing familiar design elements. In the interview with the end-users, it was mentioned that either importing data from the directory or to import data directly from the DTU always is the first procedure when using MGSS. For this reason, the starting view of the new GUI supports these functionalities – either the user can choose to import from the directory by using the import function or the user can use the notification panel to the right in Figure 4.25 to import cycles directly from the DTU. Furthermore, the starting view presents cycles that previously has been confirmed, evaluated and/or exported in the completed list. The information presented with each cycle in the completed list only shows essential information. However, the user can choose to expand this information if desired by interacting with the small arrows located next to each element. From this view, the user can shift between different dashboard layouts by interacting with the buttons in the bottom left corner. These layouts are supposed to be pre-defined by an admin user to suit different user needs and subsequently, work flows. This allows different users or groups of users to create and use their specific set of functions and lists, to facilitate an effective and standardized way of working. Like today’s MGSS, this GUI contains a familiar menu bar from where the user can perform specific actions such as open and save cycles. Additionally, the status bar from today’s MGSS has been implemented to assist the user in interpreting database and DTU status. If the user interacts with the notification panel to the right in Figure 4.25, the panel will expand. From this panel, the user can configure system settings and make profile settings.

If a DTU has been inserted into the computer, the GUI will detect this by showing a message in the status bar as well as giving the user a notification in the notification panel, as presented in Figure 4.25. If the user clicks the icon at the top of the notification panel, it will expand to the left which reveals additional information on active background processes. As shown in Figure 4.26, the user gets presented with information related to the insertion of the DTU – a timestamp and a progress bar indicating that the information on the DTU is being prepared to be used. When the progress bar in the notification panel has been filled, the user can import cycles from the DTU into the GUI. This import window is like the import window in today’s MGSS in that it provides information on the progress of the import process, it presents errors and warnings if there is a problem with a file that is being
imported. In the import view, the user gets information on each cycle that is being imported and gets feedback if a problem occurs or if the process is finished. When the import has been completed, the user can press the finish button.

![Figure 4.26 – The notification panel and import window of the prototype](image)

When the cycles from the DTU have been imported, the GUI assists the user in classifying and filtering the cycles based on fault events and fault codes, as mentioned earlier – placing cycles with familiar, non-critical fault events in the acceptable list, cycles with critical, unfamiliar fault events in the analyse list already evaluated and exported cycles in the completed list. An example of this can be seen Figure 4.27. Additionally, the GUI notifies the user that an action has been completed by placing a notification icon on the upper, right-hand side of the GUI. This notification panel can, as mentioned earlier, also be expanded to give the user information on occurred events and performed actions. The GUI presents three lists containing numerous interactive elements. Every element in the list represents a cycle with an icon, indicating cycle type. In Figure 4.27 the cycles are also represented with a time stamp and the name of the cycle. If the user needs more specific on which to base analytical decisions, expandable windows containing additional information can be obtained by interacting with the small arrows placed next to each cycle. In this example, shown in Figure 4.27, the additional information from the cycle in the acceptable list states that no fault events or fault reports were detected on the specific cycle number. From this view, the user can add cycles manually as well as add and/or customize lists to better suit a specific work flow or to create a more desirable overview. By clicking the Add Cycle icon, a menu will appear where a cycle created by adding information to that specific cycle, manually. The same principle applies to the Customize list icon, which can be seen in Figure 4.27. The design of these functions has been created in a similar way.
All elements in the lists presented earlier can be selected individually or collectively to perform further actions within the GUI. When a selection has been made, the GUI assists the user by emphasizing suitable actions and/or tools in which the selected cycle can be processed. This functionality is demonstrated in Figure 4.28. The GUI guides the user by highlighting available tools to the selected
cycles whilst hiding unobtainable tools, i.e. if a cycle in the analyse list is selected the GUI presents several analytical functions whilst hiding the export function. When a cycle in the analyse list have been selected, the user can choose any of the available tool to analyse or troubleshoot the cycle, e.g. Fault Report or MFI as exemplified in Figure 4.28. If a tool is clicked while a cycle has been selected, a new tab will open containing cycle specific information to further assist the user in performing actions. Additionally, if several cycles have been selected and a tool is opened, multiple tool tabs will open containing different data.

The fault report window is one of the most important and most used functions in MGSS. For this reason, the important information and functionalities as mentioned by users, i.e. fault events, fault codes and fault reports, have been emphasized and more easily obtained in the prototype by minimizing the number of user interactions needed to reach the information. In this prototype, the most important views are presented together, instead of in different tabs or windows, as presented in Figure 4.29. This serves two purposes; giving the user the ability to compare fault events and fault reports without having to interact with multiple windows and furthermore to give the user a simple as well as holistic overview of the most important information of this function. However, if the user wants to compare multiple fault reports on different cycles simultaneously, this can be achieved by dragging and dropping the tabs to open a new window, only containing the specific information from that specific tab.

When working with MGSS today, the most troublesome and difficult task is to interpret fault events and fault codes. By implementing the AMP into the prototype, obtainable by clicking the AMP button as shown in Figure 4.29, the user can more easily find and search for specific fault reports. Today, the user needs to open the AMP in a separate software, making fault report interpreting inefficient and cumbersome. By integrating the AMP into the prototype, fewer user interactions are needed to interpret fault reports whilst all information related to the work flow of the user is gathered in the same interface. Additionally, the fault events and fault codes can be interacted with in this view, as exemplified in the bottom right view of Figure 4.29. This way, the user can obtain additional information on faults or events that can be used in analysing or troubleshooting a specific cycle. This

Figure 4.29 – The fault reports tab of the prototype
information should be based on information in the AMP and could be presented as suggestions instead of definitive facts. The source of the information needs to be presented along with the information, whilst emphasizing the fact that the information is a suggestion. This makes the automatically generated information more transparent whilst improving the level of trust the user has for the product.

All functions available in MGSS can be reached the main view by clicking the all functions tool, located to the right in Figure 4.28. Within this tool, the user can customize the dashboard by adding additional tools or run tools directly, without adding them to the dashboard. This is exemplified in Figure 4.30. Furthermore, by implementing an information box that explains the specific functions assists the user in adding or running the correct analytical tool.

![Figure 4.30 – The function explorer in the All functions tool](image)

### 4.4.3 Testing

The testing of the proof-of-value prototype was conducted as an assisted evaluation process in a context related environment. The session was held for two hours, at a conference room at Saab AB. This testing session was conducted thirteen days after the proof-of-product testing session. All participants in the primary as well as the secondary focus group participated in the test. The testing was carried out to explore the usability of the prototype by analysing task processes, based on predefined use cases. This was conducted through observation of user interaction with the prototype, as well as by using the *System Usability Scale* questionnaire (Brooke, 1996).

Initially, the participants were, individually, presented with four isomorphic use cases that represented the users’ main goals as described in the persona. Based on the use cases the participants were asked to try and perform actions by interacting with the prototype. The participants were all asked to start each use case from the desktop, i.e. the starting view of the prototype. The use cases were described as:
1. Import one sequence of cycles from the DTU and then, add the FEA tool to the dashboard.

2. Import the file 394.86-2018-04-10 15:50:02 and find out when the cycle Engine 393.261 was exported.

3. Perform any export on all the imported cycles.

4. Find the suggested adjustment to the fault report of Flight 393.261, read it out loud, then confirm and save the fault report.

When one participant performed the use cases, the others were asked to leave the conference room to not be influenced by each other’s actions and solutions. The participant sat at a table beside the facilitator and the prototype was presented on a TV monitor. Each participant was given fifteen minutes each to perform the presented tasks, using only a computer mouse. Based on the participant’s interaction, e.g. selections and clicks, the prototype altered by introducing new interactive or non-interactive elements, dependent on the systems intended reaction. While performing the use case, the participants were asked to think out loud and explain the respective actions. The facilitator only assisted the participant if difficult problems occurred during the test.

The amount of interactions performed by the participants to complete each task were noted on a piece of paper to determine the amounts of clicks used by each participant to successfully complete a task. The task performance was then achieved by comparing the mean of the participants’ number of interaction on each task with the intended minimum amount of interactions needed to complete each task successfully. To be able to compare the performance efficiency of the prototype with MGSS, the minimum amount of interactions needed to complete similar tasks in MGSS was also determined.

4.4.4 Final evaluation

After the tests were completed, the participants of both focus groups were asked to fill out the System Usability Scale, SUS, questionnaire to measure the usability of the new GUI (Bangor, Kortum, & Miller, 2008; Brooke, 1996). Additionally, the same questionnaire was used to measure the usability of the existing MGSS to be able to compare the results of the questionnaires. Based on the participants’ answers, a total score ranging from 0-100 that corresponds to the usability of the interface was measured (Brooke, 1996). The mean SUS-score rating for both the prototype and MGSS were then described by a corresponding adjective rating (Bangor, Kortum, & Miller, 2009). This adjective was used to help the readers of this thesis interpret the results of the final evaluation. The SUS score of the prototype received a mean, M=81 (n=5, Standard Deviation, SD=8.2) and MGSS received of M=41 (n=5, SD=19.2), Figure 4.31. Consequently, the corresponding adjective rating of the prototype was closer to excellent (M=85.5, SD=10.4) than good (M=71.4, SD=11.6). The adjective rating for MGSS was closer to poor (M=35.7, SD=12.6) than OK (M=50.0, SD=13.8). The confidence interval, CI was 95%. 
The minimum amount of actions needed to successfully complete the tasks described earlier were six for the first task, six for the second, eight for the third and seven for the fourth and final task. Similarly, the minimum amount of actions in MGSS were 13 for the first task, 15 for the second, 17 for the third and 18 for the fourth task. In the last task, the amount of actions needed to troubleshoot an event using the AMP were not considered – this would increase the number substantially. The participants’ results per task can be seen compared to the minimum amount of actions needed in Table 4.4. The participant’s used 19% more actions than needed to complete tasks, on average. Using the prototype to perform tasks, the amount of actions needed to complete tasks corresponded to 43% of the minimum amount of actions needed, using MGSS.

Table 4.4 – Mean user actions per task using the prototype and MGSS

<table>
<thead>
<tr>
<th>Task</th>
<th>Mean user actions when performing tasks, using the prototype [n]</th>
<th>Minimum amount of actions needed to complete tasks, using the prototype [n]</th>
<th>Minimum amount of actions needed to complete a similar task, using MGSS [n]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>6,8</td>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>Task 2</td>
<td>7,2</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>Task 3</td>
<td>9</td>
<td>8</td>
<td>17</td>
</tr>
<tr>
<td>Task 4</td>
<td>9</td>
<td>7</td>
<td>18</td>
</tr>
<tr>
<td>Total</td>
<td>32</td>
<td>27</td>
<td>63</td>
</tr>
</tbody>
</table>

Design Conclusion

At this stage, the participants of the focus groups suggested improvements to be made to the prototype. However, since the mean SUS-score implied that the design of the developed prototype has a higher usability than MGSS, the aim of this thesis was considered achieved. Further design improvements were concluded to be implemented in future studies.
5 Discussion

In this section, the method used to develop the prototype will be discussed. Furthermore, the result of each prototyping session in relation to the questions mentioned in chapter 1.2 will be discussed. Finally, ethical aspects related to this thesis will be discussed.

5.1 Method

This section has been divided into two parts: a discussion on the methods used in the pre-study phase of this thesis and a discussion of the methods used in the design process. The method discussion relates to information covered the theoretical framework of this thesis, described in chapter 2.

5.1.1 Pre-study

One of the main focuses of this thesis has been to explore and analyse users’ needs and requirements. Due to time limitations this has not been done extensively enough to represent all the different users of MGSS. Desirably, additional interviews and/or observations should have been conducted with users of different backgrounds and working environments to gather more information related to users’ needs, goals and requirements. Even though more than a single user’s knowledge of the system has been considered by integrating insights from previous work, a more extensive user analysis would have been beneficial to be able to accumulate more relevant knowledge and collaboration to be used in the design process. Additionally, this would have given a more thorough understanding of the users, making the users more committed to the development of MGSS and subsequently, more likely to accept the prototype. However, by analysing the identified needs and integrating knowledge and insights from the NADI-model and then applying this knowledge to the persona, deep hidden user needs as described in the persona and their relation to usability could still be explored.

The benchmarking study was not a major focus of this thesis and was for that reason not conducted extensively. With a more extensive study, additional sources of information could have contributed to a better understanding of the users’ needs and requirements and how a design could meet these needs. Although the contributions of this study might not seem major, important design ideas were explored using this method. This lead to the development of early prototypes which was helpful in exploring and communicating early design ideas as well as in understanding the users’ needs and requirements. Furthermore, the benchmarking study as well as the early prototypes provided feedback at early stages of the design process which made it possible to understand the complexity of MGSS, related to users’ needs and requirements.

5.1.2 Design Process

Several major and minor usability problems was effectively identified by constructing prototype testing sessions. The different sessions were based on the intended use of the different prototypes in relation to the aim of the different design evaluation levels – proof-of-context should be used to explore and understand context of use and the participative evaluation aims to understand the user and explore design ideas, as presented in chapter 2.1.2 and 2.2 respectively. However, due to the limited time provided to conduct each testing sessions all tests had to be simplified. This resulted in narrow understanding of problems related to users’ needs and requirements in a context related environment, subsequently creating a less accurate design decisions making process. Preferably, the testing sessions should have been conducted for several hours and with additional participants, especially additional end-users. By using only one end-user in the testing sessions, the risk of the participant being affected by personal biases was high. Three end-users were supposed to take part in every testing and evaluations session but this was not possible due to unanticipated work load. If several end-users participated in the testing and evaluation session, the risk of personal bias would be substantially decreased. Additionally, if additional participants, both end-users, surrogate and novices, had been
Discussion

included, a deeper understanding of the problems related to context of use could have been accumulated in the design of the prototype. Additional participative users in the design process would also have implied additional user experience, characteristics and abilities limitations which in turn would have enhanced the usability further.

A recurring problem with the testing sessions was that the experienced participants, with whom the different tests were conducted, could not mentally stand back from details and specifics. This contradict the information presented by Cross (2004) and Ho (2001). By being even more specific when presenting the aim of each testing session and interrupting the discussion when it became too focused on details, this problem could have been avoided.

Since the participants of the focus groups lacked design experience whilst not being familiar with design testing methods, a simple and familiar voting system was used to determine the potential of each prototype in relation to the usability criteria described in 1.2 Aim and purpose. This way, the limited time of each testing sessions could be effectively utilized whilst providing valuable information on which to make design decisions. However, all testing sessions were conducted by a single facilitator. Conducting usability tests where same person is the note-taker and the “computer” makes it difficult to capture what the participant says or does. Instead, these roles should have been divided amongst several individuals to make sure not to lose valuable information as well as to facilitate the data analysis process. The possibility of recording the audio of the testing sessions was considered, but due to security policies on integrated audio-visual technology, this was not possible.

The troublesome situation mentioned above was especially apparent at the second stage of the design process, the proof-of-concept prototyping. This assisted evaluation test was conducted to observe user interaction whilst testing learnability and efficiency. Having numerous goals with one single usability test was proven to be challenging – the situation got hurried, due to the time limit of the testing session; participant interactions and thoughts were difficult to capture, due to the use of a single facilitator; and for the same reason, testing learnability and efficiency by capturing unintended interactions and time to task was difficult. Additionally, the prototypes were presented in the same order to all the participants. This affected the impression of the first two prototypes negatively and the last three positively, which corresponds to the distribution of the participants’ votes. To increase the validity and subsequently, credibility of the result from this prototyping session, the testing session should have been conducted as three separate usability tests: one that aims to observe user interaction, one that aims learnability and another that aims to test efficiency. Also, to not affect the participants’ votes, the order in which the prototypes were presented should have been altered.

A controlled evaluation would have been appropriate to use to truly determine user performance success whilst increasing the validity of the prototype. In this type of evaluation, usability can be tested more extensively by e.g. a comparison of time to task with the prototype and MGSS. The developed prototype does not illustrate all possible use cases and therefore not replicating the real working environment in an acceptable manner, making a comparison biased. Instead, quick and dirty methods were utilized to get an impartial indication of the usability of the prototype.

When comparing the minimum amount of actions needed to complete tasks in the developed prototype as well as completing similar tasks in MGSS, the actions were performed by the author of this thesis instead of an experienced user. This especially affects the amount of actions needed to perform tasks using MGSS. Even though the specific user interactions performed to complete tasks was based on and inspired by walkthroughs in MGSS as well as by observing the daily usage of the end-user of MGSS, the minimum actions needed using MGSS could possibly be reduced further. However, since the amount of actions related to usage of the AMP was not considered the amount of actions described in Table 4.4 can be considered reasonable.
The decision to use design evaluation, as mentioned in chapter 2.2, was shown to be an effective way of identifying and solve usability related problems. By taking a user-centred and qualitative approach when evaluating the prototypes, valuable feedback on the design was achieved early in the development process. Even though the analysis process related to the evaluation sessions was time-consuming, this created a better understanding of the problem related to usability, subsequently leading to an improved prototype. However, several changes to evaluation process could have been made to improve the prototype in terms of usability – the design evaluation process was primarily conducted with experienced or surrogate users, whom all were personally biased towards design decisions, resulting in replicable results. This affected the participative evaluation and the suggested improvements of each evaluation session negatively – the participants got stuck in specifics and details as mentioned earlier. Instead, to effectively identify genuine and fundamental problems in the design of MGSS, novices should also have been included and their design ideas should have been considered in the design evaluation process. This would expand the design exploration by consideration of additional ideas as well as be valuable in understanding fundamental design problems within MGSS.

5.2 Results

In this section, the results from the proof-of-value prototyping session will be discussed. The result discussion is related to information covered the theoretical framework of this thesis, described in chapter 2.

5.2.1 Graphical user interface design

Previously, it has been stated that the user wants to work in a process oriented way whilst making fault analysis by using simple and effective tools. Furthermore, the user wants control by getting a simple overview in a flexible, understandable and easy-to-use interface. The prototype has been developed based on these needs and requirements as well as by interpreting insights from all the previous testing and evaluation sessions into design elements, to more accurately enhance the usage of MGSS.

The design of the prototype is based on a customizable but simple dashboard application that supports multiple user needs and requirements. The dashboard has been inspired by software familiar to the end-users, making the interface more easily understandable. By implementing a simple and process oriented work flow, in which the user gets guided and assisted in anticipating the next action the user can more easily and effectively interpret fault events. The prototype lets the user modify and customize the interface to some extent whilst being able to perform actions in several different ways, creating a flexible working environment.

The starting view has been designed with the aim to decrease cognitive strain by using a clear, understandable overview containing familiar design elements. This helps the user in making choices to more effectively progress the analytical work process. Additionally, the starting view presents cycles that previously has been confirmed, evaluated and/or exported in the completed list. This gives the user a sense of control as well as a simple overview. All cycles in this prototype are represented with an icon, indicating cycle type, and with a time stamp. This way, the information related to each cycle will not overwhelm the user but instead, assist the user in more effectively interpreting fault events by making the information more easily obtainable. Additionally, this functionality aims to give the user a more perspicuous overview where only essential information is presented.

The prototype guides the user by highlighting available elements, e.g. if a cycle in the analyse list is selected the GUI presents several analytical functions whilst hiding the export function. This functionality assists the user in performing the correct tasks, decreasing faults and errors related to usage. Furthermore, this helps the user in making choices to more effectively progress the analytical work process. Interactive as well as static elements within the prototype that share similar purposes
have been created in a similar way, making the GUI more consistent and easier to understand and subsequently, easier to learn.

All elements in the lists can be selected individually or collectively to perform further actions within the GUI. This an important feature needed to support a flexible working process. Additionally, if several cycles have been selected and a tool is opened, multiple tool tabs will open containing different data. This functionality makes the process of comparing e.g. fault reports or variables possible, which is an important feature of MGSS today.

From all different views within the prototype, the user can reach the notification panel. From this panel, the user can configure system settings and make profile settings. This panel also aims to help the user have control and get a simple overview of work flows and processes working in the background of the system. Today, the user gets presented with information related to the insertion of the DTU – a time stamp and a progress bar indicating that the information on the DTU is being prepared to be used. This process locks the system from any other action until it is completed. By letting time-consuming processes, such as the preparation of the DTU, work in the background the user can proceed with their routine, e.g. set up or change dashboard layout, add cycles or get an overview of previously evaluated cycles. Being able to complete work without getting locked out by inefficient processes aims to optimize the user’s work flow, leading to a more efficient and flexible analysis process.

In MGSS today, it is a difficult task to understand what function to use and why. Additionally, the desired function is most likely located in the extensive drop-down menus that exists throughout the system. By utilizing a search function and by organizing functions by name, the process of finding and using a specific function can be more effectively utilized. Furthermore, by being able to run tools directly from the all functions view, without adding function to the dashboard, the user can keep the dashboard clean and simple whilst performing analytical tasks.

The prototype utilizes feedback and error boxes when the user is performing tasks more extensively. This is important to make automated processes transparent, subsequently making the result of these processes acceptable and trustworthy, from the user’s perspective. This type of transparency assists the user in understanding problems which in turn improves the user experience of the GUI.

### 5.2.2 Evaluation

In this section, notable results of the various prototyping evaluations will be discussed.

**Proof-of-concept**

The amounts of occurring errors in the Clean Setup and Adaptive Dashboard were significantly higher than the other concepts, this could imply that the concept were less intuitive and harder to understand. This could also be an effect of the fact that the prototypes were presented in the same order to all the participants.

**Proof-of-product**

When concluding the proof-of-product prototyping session, several changes to the prototype as suggested by the focus group were decided to not be implemented in the final prototype, e.g. implementing different icons or making the progress bar 100%. These changes were concluded to be minor changes with no apparent effect on the usability of the prototype at this stage. If all details and specific elements and functions would be implemented in the design of the final prototype, the desired evolutionary classification of the prototype would be lost. This would in turn affect the way the prototype is interpret and evaluated.
Proof-of-value
The mean SUS-score of the prototype was significantly higher than in the existing MGSS, implying that the design of the developed prototype has a higher usability than MGSS. Based on the SUS-score, the usability of the prototype was excellent, whereas the usability of MGSS was poor. Additionally, the standard deviation of the prototype (SD = 8.2) indicates a consensus in terms of usability. This consensus relates to the homogeneity of the participants in terms of user experience, characteristics and limitations. The standard deviation of MGSS was greater (SD = 19.2). This result was severely affected by the fact that the experienced participants felt confident using the existing system. This was rather expected, since the participants all use MGSS in their daily work in one way or another.

The task performance indicated that the participants were successful when performing the predefined use cases. This is reflected by the mean user actions needed to complete each task, compared to the calculated minimum actions needed to complete the tasks, using the prototype. In total, the participants only used 19% more interactions than needed. This implies that the prototype is understandable and easy to use. Additionally, as mentioned in section 4.4.4, the amount of actions needed to complete tasks using the prototype corresponded to 43% of the minimum amount of actions needed using MGSS. This indicates that the prototype can more effectively assist the user in performing tasks, implying an improved efficiency.
6 Conclusion

This thesis achieved the aim by demonstrating how prototyping and a user-centred design approach was used to develop a maintenance support system prototype. The final evaluation of the prototype indicated a substantial increase in usability as well as an efficiency improvement when performing tasks, in comparison to the existing system.

The amount of actions needed to complete a task has been decreased by creating a prototype that uses a simple interface to support process and task oriented work flows. By creating an interface that assists the user in interpreting fault events whilst providing feedback, the efficiency on analytical tasks can be improved. The prototype provides guidance, directing the user in performing the correct tasks, which effectively improves the efficiency of the work flow by increasing the control and confidence of the user.

The developed prototype enhances the usage by only presenting the essential information needed to perform fault event analysis, this gives the user a simple overview. By utilizing the key principles of gestalt systems in a simple and modern design approach, the interface becomes consistent and understandable whilst increasing the cognitive ease. The dashboard resembles familiar software to the end-users, making the interface more easily understandable which enhances the usage.

The prototype lets the user modify and customize the interface whilst being able to perform actions in several different ways, creating a flexible working environment suitable for many different user needs, goals and requirements. By implementing tabs instead of windows as navigation element, the user can easily and flexibly get access to and compare essential information.

The interface has been made comprehensible, easier to understand and learn by utilization shallow interaction levels. Abundant information has been replaced by essential information through symbols and a few lines of text, making the interface consistent and clear. By preventing user errors, providing feedback on success and failure as well as making automated processes transparent, the prototype assists the user in learning the system.

6.1 Future research

This thesis illustrated how design could be used to improve the usability of MGSS, but the prototype is still to be considered as an evolutionary prototype. Additional and more extensive usability testing is needed to improve the usability of the prototype. Preferably, this is done by further developing the prototype in a user-centred development process, like the one described in this thesis.

Some of the usability characteristics described in this thesis tend to contradict each other, e.g. flexibility and efficiency – a customizable interface assists the user in performing specific actions in several different ways, but not necessarily in the most efficient way. It would be interesting to study usability characteristics and their respective contribution to the usability of GUIs. Additionally, by quantifying usability characteristics the most efficient way of improving usability could be determined.
7 Reference List


Rick Johnson

Demographics
- Male, 42 years
- Linköping, Sweden
- Flight test engineer
- Married
- "My experience is that all parties have something to learn from exchanging knowledge"

Background
Rick is an flight test engineer at Saab AB, working with MGSS on a daily basis. He is responsible for data analysis in MGSS. Rick’s daily routine includes importing flight data from different DTUs, analyzing cycles and troubleshooting unfamiliar fault events.

Rick is not a highly trained computer analyst, quite the opposite. He is not used to handling computerized, analytical systems. Still, he does most of his work alone.

Scenarios
Rick wants to be able to quickly get a simple overview of data and, by help of smart tools and functions, effectively analyze these data. He wants to be able to organize his data, this gives him the sense of control over the situation. Additionally, he says it is crucial to be able to do internal and external exports.

Solutions
Rick wants to work in a familiar software environment where it is possible to:
- Have the ability to see information from specific aircrafts;
- Clearly see fault events from a cycle;
- Be able to get a simple overview;
- Export data through different exports;
- Get feedback on which exports has been done and when;
- and be able to create a cycle manually.

Rick needs to work on a windows based computer platform, with the windows application MGSS installed to be able to carry out his work. To facilitate his daily work, he wants two computer monitors and his cheat sheet containing the most frequently occurring fault codes. The better keep track of completed work and to better communicate his work with his co-workers he wants a whiteboard.

Goals
Rick needs to analyze flight data to keep the aircrafts operating at a low cost and with high sustainability by:
- Transferring data into MGSS;
- Making fault analysis;
- Troubleshooting unfamiliar fault events;
- and by exporting data to other systems.

Themes
When working with MGSS Rick wants to share information and to have control when doing so. But, Rick needs assistance.

Expectations
With the new MGSS graphical user interface Rick expects:
- Improved efficiency to decrease cost while maintaining high sustainability;
- Enhanced usage through a simple user interface containing easy-to-use functions;
- Improved learnability by making the user interface familiar and understandable;
- and increased flexibility by making the user interface adaptive