Design and Evaluation of a User Interface for a Program, Used in a Flight Test Environment of Electronic Warfare Systems, with Regards to Usability in Terms of Low Error Rate

Design och utvärdering av ett gränssnitt för ett program som används i samband med flygprov och testning av plans EW-system med fokus på användbarhet och låg felfrekvens

Elin Larsson

Supervisor : Anders Fröberg
Examiner : Erik Berglund
Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnt som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är krävande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/.

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/hers 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/.

© Elin Larsson
Abstract

In today's society, the amount of connected devices is increasing. Many of these devices are in need of an interface to enable interaction with users. These devices are built for solving tasks of varying degree of difficulty and the need for user-friendly interfaces are highly relevant. **Errors** is one attribute that needs to be considered when creating a user interface. Therefore, this thesis project investigates how one can design and evaluate a user interface, which is used in a flight test environment, with regards to usability in terms of low error rate. Based on requirements gathered in the requirements elicitation process together with inspiration from the anchor-based subgoaling design principle, a low-fidelity and a high-fidelity prototype is created. To evaluate these prototypes user testing together with the thinking aloud technique is used. Data regarding the number of errors made by each user, if they complete a task or not and the users general thoughts of the prototypes are also gathered. The whole process of implementation and evaluation is done in an iterative manner. The results show that the users are satisfied with the interface, complete or partially complete each task, make few errors and can recover from the mistakes they make. Based on the results, a conclusion from this thesis can be drawn that by using an iterative design, creating prototypes inspired by the anchor-based subgoaling design principle and conducting user tests together with the thinking aloud technique one can create and evaluate a user interface with regards to usability in terms of low error rate.
Acknowledgments

I would like to thank my supervisor at FMV for supporting me through this thesis project and all the colleagues at FMV for filling the past five months with a lot of joy and laughter. Thanks to my supervisor and examiner at Linköping University for guiding me in the right direction. Finally, I would like to thank my family and my boyfriend for encouraging me and supporting me all the way.

Elin Larsson
Linköping, December 2019
Contents

Abstract iii
Acknowledgments iv
Contents v
List of Figures vii
List of Tables viii
1 Introduction 1
  1.1 Aim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
  1.2 Research question . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
  1.3 Delimitations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
2 Background 3
3 Theory 5
  3.1 Usability . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
  3.2 Usability Engineering . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
  3.3 Requirements Elicitation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
    3.3.1 Interviews . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
    3.3.2 Prototyping . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
  3.4 Usability Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
    3.4.1 Usability Testing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
      3.4.1.1 Prototyping . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
    3.4.2 Thinking Aloud . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
    3.4.3 Validity and Reliability . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
  3.5 Errors . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
    3.5.1 Error Prevention . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
4 Method 17
  4.1 Structure of the Study . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
  4.2 Pre-study . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
    4.2.1 Requirements Elicitation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
  4.3 Implementation and Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
    4.3.1 Implementation: Low-fidelity Prototype - Iteration 1 . . . . . . . . . . . 20
    4.3.2 Evaluation: User Testing - Iteration 1 . . . . . . . . . . . . . . . . . . . . . 20
    4.3.3 Implementation: High-fidelity Prototype - Iteration 2 . . . . . . . . . . . 22
    4.3.4 Evaluation: User Testing - Iteration 2 . . . . . . . . . . . . . . . . . . . . . 22
5 Results 23
  5.1 Pre-study . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Illustration of the main purpose of the program.</td>
<td>4</td>
</tr>
<tr>
<td>5.1</td>
<td>Starting page for the low-fidelity prototype.</td>
<td>27</td>
</tr>
<tr>
<td>5.2</td>
<td>Parallel view for creating a new test session.</td>
<td>28</td>
</tr>
<tr>
<td>5.3</td>
<td>The page representing a started test session.</td>
<td>28</td>
</tr>
<tr>
<td>5.4</td>
<td>Starting page for the high-fidelity prototype.</td>
<td>31</td>
</tr>
<tr>
<td>5.5</td>
<td>The last page presented when creating a new setup.</td>
<td>32</td>
</tr>
<tr>
<td>5.6</td>
<td>The first page when creating a new session.</td>
<td>32</td>
</tr>
<tr>
<td>5.7</td>
<td>The third and final page when creating a new session.</td>
<td>33</td>
</tr>
<tr>
<td>C.1</td>
<td>Paper prototype for the requirements elicitation - first view.</td>
<td>52</td>
</tr>
<tr>
<td>C.2</td>
<td>Paper prototype for the requirements elicitation - second view.</td>
<td>53</td>
</tr>
<tr>
<td>C.3</td>
<td>Paper prototype for the requirements elicitation - third view.</td>
<td>53</td>
</tr>
<tr>
<td>E.1</td>
<td>Low-fidelity prototype for iteration 1 - start page.</td>
<td>57</td>
</tr>
<tr>
<td>E.2</td>
<td>Low-fidelity prototype for iteration 1 - second page.</td>
<td>58</td>
</tr>
<tr>
<td>E.3</td>
<td>Low-fidelity prototype for iteration 1 - third page.</td>
<td>58</td>
</tr>
<tr>
<td>E.4</td>
<td>Low-fidelity prototype for iteration 1 - fourth page.</td>
<td>59</td>
</tr>
<tr>
<td>E.5</td>
<td>Low-fidelity prototype for iteration 1 - fifth page.</td>
<td>59</td>
</tr>
<tr>
<td>E.6</td>
<td>Low-fidelity prototype for iteration 1 - sixth page.</td>
<td>60</td>
</tr>
<tr>
<td>E.7</td>
<td>Low-fidelity prototype for iteration 1 - seventh page.</td>
<td>60</td>
</tr>
<tr>
<td>E.8</td>
<td>Low-fidelity prototype for iteration 1 - eighth page.</td>
<td>61</td>
</tr>
<tr>
<td>E.9</td>
<td>Low-fidelity prototype for iteration 1 - ninth page.</td>
<td>61</td>
</tr>
<tr>
<td>E.10</td>
<td>Low-fidelity prototype for iteration 1 - tenth page.</td>
<td>62</td>
</tr>
<tr>
<td>E.11</td>
<td>Low-fidelity prototype for iteration 1 - eleventh page.</td>
<td>62</td>
</tr>
<tr>
<td>E.12</td>
<td>Low-fidelity prototype for iteration 1 - twelfth page.</td>
<td>63</td>
</tr>
<tr>
<td>E.13</td>
<td>Low-fidelity prototype for iteration 1 - thirteenth page.</td>
<td>63</td>
</tr>
<tr>
<td>E.14</td>
<td>Low-fidelity prototype for iteration 1 - popup windows and minor objects.</td>
<td>64</td>
</tr>
<tr>
<td>F.1</td>
<td>High-fidelity prototype for iteration 2 - start page.</td>
<td>65</td>
</tr>
<tr>
<td>F.2</td>
<td>High-fidelity prototype for iteration 2 - new session: first page.</td>
<td>66</td>
</tr>
<tr>
<td>F.3</td>
<td>High-fidelity prototype for iteration 2 - new session: second page.</td>
<td>66</td>
</tr>
<tr>
<td>F.4</td>
<td>High-fidelity prototype for iteration 2 - new session: third page.</td>
<td>67</td>
</tr>
<tr>
<td>F.5</td>
<td>High-fidelity prototype for iteration 2 - replay session: first page.</td>
<td>67</td>
</tr>
<tr>
<td>F.6</td>
<td>High-fidelity prototype for iteration 2 - replay session: second page.</td>
<td>68</td>
</tr>
<tr>
<td>F.7</td>
<td>High-fidelity prototype for iteration 2 - new setup: first page.</td>
<td>68</td>
</tr>
<tr>
<td>F.8</td>
<td>High-fidelity prototype for iteration 2 - new setup: second page.</td>
<td>69</td>
</tr>
<tr>
<td>F.9</td>
<td>High-fidelity prototype for iteration 2 - new setup: third page.</td>
<td>69</td>
</tr>
<tr>
<td>F.10</td>
<td>High-fidelity prototype for iteration 2 - change setup: first page.</td>
<td>70</td>
</tr>
<tr>
<td>F.11</td>
<td>High-fidelity prototype for iteration 2 - change setup: second page.</td>
<td>70</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>List of participants in the requirements elicitation process.</td>
<td>18</td>
</tr>
<tr>
<td>4.2</td>
<td>List of participants in the user tests in iteration 1.</td>
<td>20</td>
</tr>
<tr>
<td>4.3</td>
<td>List of participants in the user tests in iteration 2.</td>
<td>22</td>
</tr>
<tr>
<td>5.1</td>
<td>Total number of times an error occurred during user testing in iteration 1.</td>
<td>29</td>
</tr>
<tr>
<td>5.2</td>
<td>Success rate for each task during user testing in iteration 1.</td>
<td>29</td>
</tr>
<tr>
<td>5.3</td>
<td>Total number of times an error occurred during user testing in iteration 2.</td>
<td>33</td>
</tr>
<tr>
<td>5.4</td>
<td>Success rate for each task during user testing in iteration 2.</td>
<td>34</td>
</tr>
</tbody>
</table>
1 Introduction

Over the last decade the technical evolution has increased rapidly. Over 2 billion\(^1\) computers were in use in the world by the year 2015 and researches estimate there will be 50 billion\(^2\) connected devices when we reach year 2020. The everyday life is striving towards more automation and the research area of human-computer interaction is gaining in relevance. In the earlier days humans interacted with computers by typing in simple commands, while today many computer programs are designed with graphical user interfaces to assist the user in solving tasks of varying degree of difficulty. However, humans and computers do not always think and behave in the same way, so the quest for interfaces designed with regards to usability is highly relevant.

Studies show how different designs and implementations of an interface can give various outcomes, where all not aligning with the purpose of the system and reaching the goal. For example, Pauwels et al \(^1\) study where colour instead of asterisks were used to mark required fields in online forms, or the study performed by Reeder and Maxion \(^2\) where the success in setting file permissions differed based on the design and detail of the user interface. One can either analyse the results as users making mistakes while using the interface, or as failure committed by the designers when not making the interface clear and intuitive enough to guide the users in the right direction.

According to Nielsen \(^3\), errors is one of the attributes to consider when designing interfaces with regards to usability. People tend to do mistakes and produce errors when being interrupted and not easily guided back to the previous action, or when tasks and procedures require them to behave in unnatural ways \(^4\). If mistakes and possible errors can be caught and detected early, or even better, prevented from happening, the risk of them leading to critical consequences can be decreased.

1.1 Aim

The aim of this thesis is to examine how you can design and evaluate a user interface with regards to low error rate, specifically for a program used in a flight test environment. The

\(^1\)https://www.worldometers.info/computers/  
1.2 Research question

To reach the aim of this thesis, the following research question has been formulated:

- How can you design and evaluate a user interface for a program, used in flight tests of electronic warfare systems, with regards to usability in terms of low error rate?

In this context, low error rate is based on the definition by Nielsen saying that: "the system should have a low error rate, so that users make few errors during the use of the system, and so that if they do make errors they can easily recover from them".

1.3 Delimitations

This study has only focused on evaluating the design on a small target group, where the test persons were assumed to have prior knowledge in the field where the program will be used. Furthermore, the design and prototype is specifically developed for using within the area of flight tests and when testing EW-systems.
This thesis is done in collaboration with FMV, The Swedish Defence Materiel Administration, which is an independent civil authority under the Ministry of Defence. The main purpose of FMV is to develop and deliver equipment to support the Swedish Armed Forces, making sure they can accomplish their missions. Before anything is delivered to the customers, FMV makes sure it is safe and well tested. Apart from that, FMV also maintains and repairs material from the Swedish Armed Forces such as planes, submarines and helicopters.

At FMV T&E in Linköping, which is one of three test and evaluation centres, they perform so called flight tests where you gather data during a flight of an aircraft. This data is analysed to evaluate the technical functionality of the aircraft and its usability in the context of use. The two main purposes of a flight test is to fix any detected design problems and to document the capabilities of the tested aircraft. A flight test can range from testing a small, specific part of a system, to the complete functionality of a newly developed aircraft.

In the context of this thesis, the user interface is designed and implemented for a program that is part of the environment for a flight test of electronic warfare systems. Electronic warfare can be defined as military activity using the electromagnetic spectrum [5]. Either to make it more difficult for the enemy to use the spectrum for own purposes, such as obtaining, processing or communicating information. Or for your own protection against enemy forces using the spectrum to destroy and tamper with your systems. In a fighter aircraft, an electronic warfare system is installed to tamper with the enemy’s radar and robots, and to minimise the risk of being detected and defeated in combat [6].

The main purpose of the program is illustrated in Figure 2. R1 and R2 are responsible of detecting and following the target and send data to F(x), which in turn will perform calculations and send the resulting data to S1, S2 and S3 to direct them and point them towards the target. The F(x) will in this thesis be called “the program” and the user interface that will be developed and evaluated is for that specific program. Furthermore, the system responsible of detecting and following the target (in this figure called R1 and R2) will in this thesis be called tracker while the other system who receives data and is being told where to point (in this figure named S1, S2 and S3) will be called slave. As seen in the figure, one tracker can be connected to one or more slaves.
Figure 2.1: Illustration of the main purpose of the program.
This chapter presents the relevant theory for understanding the context and workflow of this thesis. First of, the term usability is explained followed by theory regarding usability engineering and requirements elicitation. Then the area of usability evaluation is described and especially usability testing with prototyping and the thinking aloud technique. Lastly the term errors is explained and some theory regarding error prevention.

3.1 Usability

A commonly used definition of usability is the one stated by the International Organization for Standardization [7] which is formulated as follows: "degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". Furthermore, the following attributes are said to compose the term usability: appropriateness recognisability, learnability, operability, user error protection, user interface aesthetics and accessibility. Apart from that, many other attempts for defining usability and the corresponding usability attributes have been made. One example is the classification done by the usability expert Jacob Nielsen [3] where he determines the five key attributes learnability, efficiency, memorability, errors and satisfaction as being the ones most widely used.

However, according to several researchers [8], [9], [10], there does not seem to exist a consensus in the usability area of how to define the term and what attributes to associate with it. Regarding the usability attributes and its ambiguity, Vince Bruno and Ghassan Al-Qaimari [9] believes that the reason behind this is the fact that the usability area has been under development, and still is. The design of interfaces have evolved from simple switches to graphical interfaces, the interaction techniques are evolving towards voice and gestures, and the targeting user groups earlier consisted of experts while today they mainly consist of the real end-users. Alonso-Ríos, Vázquez-García, Mosquera-Rey and Moret-Bonillol [8] adds to this discussion, regarding the lack of consensus in the usability area, by saying that the existing definitions are too brief and ambiguous. With more details it would be easier to perform usability studies in the real world.

Vince Bruno and Ghassan Al-Qaimari [9] reach a conclusion saying that at least four common factors need to be considered when developing a system with regards to usability: knowing the target user group, knowing the system specific tasks that need to be implemented, understanding the technology and its possible limitations, and considering the
context in which the system will be used. Based on these factors, usability attributes can then be chosen with the best fit for the specific context in mind. Alonso-Ríos et al \[8\] states that even though there is a plethora of usability attributes to analyse when doing a usability study, there is no demand for using all of them. Every system has their own specific characteristics and thus all usability attributes will not always be the best fit and does not have to be considered.

3.2 Usability Engineering

Usability engineering is an area connected to human-computer interaction and with the focus on the process of developing and implementing usable user interfaces. According to Nielsen \[3\] the usability engineering lifecycle consists of the following stages:

- **Know the user** - study the intended users and how they will use the product. Involve not only the end users in this investigation, but everyone that could be affected by the product.

- **Competitive analysis** - use existing similar products and observe users while testing them. Take notes on strengths and weaknesses and use the information to build a better and improved product.

- **Setting usability goals** - prioritise the usability aspects and determine the acceptance level for measurements of the relevant usability attributes.

- **Parallel design** - have several designers early working on different approaches. Choose the best design amongst the alternatives which you further develop.

- **Participatory design** - involve users early in the design process and encourage them to ask questions and share own thoughts and ideas.

- **Coordinated design of the total interface** - make sure the usability characteristic consistency is not violated. Regarding all parts of the product from application screens, to documentation and online help systems.

- **Apply guidelines and heuristic analysis** - all projects should either follow the general guidelines for user interface design, or more category specific guidelines if applicable. These guidelines can then be used as the base for an heuristic evaluation.

- **Prototyping** - the idea with using prototypes for user testing is to save time and cost. A prototype can be changed and refined several times, based on input and feedback from users, before implementing the final product.

- **Empirical testing** - you will benefit more from just choosing any of the existing usability engineering methods to test the product and discover usability problems, than if you try to find the exactly right method for your specific project.

- **Iterative design** - one important reason to combine evaluation and iterative design is because a solution that might solve one usability problem might instead introduce another. By an iterative design you can refine the product several times, get feedback and try to create a final product meeting the requirements.

- **Collect feedback from field use** - a newly released product, its users and the users behaviour can be analysed and used as a starting point for future products.

Furthermore, Nielsen points out that the usability engineering process can be successful even though not all the stages are completely fulfilled. But the important thing is not to rush into the design phase and instead put a lot of effort in the preparing stages.
3.3 Requirements Elicitation

One of the most important and complex part of a software development process is the requirements elicitation. It is usually performed in the beginning of a project and its purpose is to uncover and gather requirements for the type of computer-based system that is to be developed. Zowghi and Coulin [11] presents a survey of the many different techniques and approaches that exist in the field, and also discusses the challenges met by practitioners and researchers.

It is widely known that requirements elicitation is a difficult process and demands for good communication skills from the software engineers as well as high commitment from the stakeholders. The most important task is not to only uncover and gather the requirements, but the right requirements. Requirements elicitation is an iterative process and how it is performed depends on the system to be developed and the aim of the project. In reality, both time and budget also have a huge impact on the process. [11]

The process of requirements elicitation can be divided into five steps and are described below [11]:

- **Understanding the application domain** - the "real world" where the system will be used needs to be thoroughly examined. Meaning both political, organisational as well as social aspects that might affect the system.

- **Identifying the sources of requirement** - it is important to have a wide perspective when searching for the requirements since they can exist in different formats and belong to different sources. The most obvious source is the stakeholders, but it is also important to use existing systems and its documentation. Users and domain experts are also potential sources good for identifying problems and user needs.

- **Analysing the stakeholders** - the stakeholders for a specific system can vary, but it usually means any one that has an interest in the system or will be affected by the development or the final product. The customer of the project is the first stakeholder that comes to mind, but it can also be other persons either internal or external to the organisation, or the actual end users.

- **Selecting the techniques, approaches and tools to use** - choosing only one elicitation technique usually does not produce good results in the elicitation process. A variety of techniques is preferable, used at different stages in the process, and the choice of certain techniques depends on the context of the project.

- **Eliciting the requirements from stakeholders and other sources** - this last step is where the actual elicitation starts, when the sources of the requirements have been identified and the appropriate techniques and tools have been chosen. The important part is to identify the actual needs and wants of the stakeholders.

When determining which techniques to use for the elicitation process, the choices are many. According to Hickey and Davis [12] the less experienced analyst tends to choose a technique either based on the fact that it is the only technique he or she knows, or because they believe a technique that worked well last time should be at least as successful this time as well. Zowghi and Coulin [11] presents domain analysis as one type of technique which is especially useful if the aim is to replace or enhance an existing system. The idea is to quickly acquire domain knowledge by examining existing documentation and applications, and possibly reuse some components or ideas. This technique can be useful as a baseline and support in creating a common understanding between the analyst and the stakeholder. Interviews and prototyping are two other example of techniques for requirements elicitation, further described down below in separate sections.

According to Zowghi and Coulin [11] some techniques are better suited for a specific requirements elicitation activity and different types of techniques can either be complementary
or alternative. Interviews and domain analysis are for example both suitable for all five ac-
tivities in the elicitation process, while prototyping is best suited for analysing the stakehold-
ers and eliciting the requirements. Furthermore, interviews and domain analysis are assumed
to be complementary techniques while prototyping is seen as an alternative technique to ei-
er interviews or domain analysis. When choosing techniques for the elicitation process, it
is advisable to pick complementary techniques to achieve the best possible results. However,
if a specific technique did not perform as good as expected, or if the analyst is unable to use
a specific one, an alternative technique might be useful.

3.3.1 Interviews

One of the most commonly used requirements elicitation techniques is interviews. Interviews
are especially useful since they provide the analyst with a lot of information and data in a
quick manner. However, the quality of the retrieved data highly depends on the level of
interaction between the participants. Interviews can be performed in three different ways:
unstructured, structured or semi-structured. An unstructured interview is best suited when
the domain knowledge is low, or in a set of several interviews where the upcoming ones are
formed in a structured manner. The unstructured interview does not follow a strict agenda
and might therefore either focus too much on details or completely miss to cover other areas
of interest. Structured interviews, on the other hand, have a predetermined agenda with a
set of questions to be asked. The difficulty is in asking the right questions, at the right time,
and one disadvantage with this approach is the lack of investigation of new ideas. A semi-
structured interview is basically a combination of the former two. [11]

In the article The role of domain knowledge in requirements elicitation via interviews: an ex-
ploratory study written by Hadar, Soffer and Kenzi [13] they perform an empirical study in
which they examine the effects domain knowledge have when performing interviews with
customers, in the requirements elicitation process. Participants in the study had either do-
main knowledge or no domain knowledge. One of the main positive aspects mentioned
by the participants with domain knowledge was the ability to formulate focused and com-
prehensive questions. While the ones with no domain knowledge thought they could have
avoided the general questions, asked more specific and detailed ones and covered all the rele-
vant areas if they had not lacked the domain knowledge. Also, the lack of domain knowledge
caused some trouble in understanding the customer and the terminology used. Regarding the
negative affects observed by the participants with domain knowledge, it was about being to
fixed with your own point of view, not asking the trivial questions and not really listening
to the customer and thus missing important information. When it comes to the possibility
of having contradicting views with the customer, the participants with domain knowledge
believed it was both a positive and a negative aspect. Either it could create a conflict or en-
courage discussion and overcome bias. The results showed that domain knowledge affects
both the communication with the stakeholder and the understanding of what the stakeholder
needs.

When performing interviews it is important to ask the right questions, retrieve the in-
formation needed and formulate correct requirements. Burnay, Jureta and Faulkner [14] in-
troduces a so-called Elicitation Topic Map (ETM) to form as a checklist when performing
interviews and also indicate how important different topics are. A topic is said to be more
important if the stakeholder is more likely to share information and talk about it, without
being explicitly asked by the interviewer. The results showed that topics regarding for exam-
ple which actors are going to use the system, where the system will be used and the purpose
of the system were most likely spontaneously shared by the stakeholder. While information
concerning the legal and financial status of the stakeholders company and trends and atmo-
sphere in the company where much less shared, unless explicitly asked by the interviewer.
Furthermore, topics such as the vision and strategy of the company were not stated as being
shared spontaneously but such information might be relevant during the requirements elic-
3.3. Requirements Elicitation

The ETM could therefore be used as an indicator for what questions to ask based on what information you want to retrieve.

Bano, Zowghi, Ferrari, Spoletini and Donati [15] conducted a study in the article *Learning from Mistakes: An Empirical Study of Elicitation Interviews performed by Novices* where they investigated the most common mistakes done by novices when performing an interview for requirements elicitation. Apart from teaching novices how to prepare for an interview by asking the right questions, it is also important to pay attention to the most commonly made mistakes. The aim of this study was therefore to provide a list containing these mistakes and examples, to improve students ability to perform an effective requirements elicitation interview. The results contained a classification of the 34 found mistakes into the following different categories:

1. Question Formulation
2. Question Omission
3. Order of Interview Questions
4. Communication Skills
5. Analyst Behaviour
6. Customer Interaction
7. Teamwork and Planning

The three most frequently tracked mistakes were: asking vague questions, incorrectly ending of interview and not building rapport with the customer. These mistakes were classified into the categories Question Formulation, Order of Interview Questions and Customer Interaction respectively. The consequences of these mistakes can lead to ambiguities, misinterpretations, making the customer feel uncomfortable and other issues further on in the development process. Other mistakes that were noticed in more than half of the cases were: asking technical questions too early, incorrect opening of interview such as no presentation of the interviewer herself nor about the project, incorrect order of questions and unnatural dialogue style where the customer had trouble understanding the questions due to poor linguistic.

3.3.2 Prototyping

Prototyping is a technique commonly used when developing human-computer interfaces and when there exist similar systems. It is a useful technique for retrieving feedback on possible solutions of the system and it is often used together with interviews. Prototyping can be performed in several different ways, by for example using storyboards, or an executable, throwaway or evolutionary prototype. The main advantage with prototyping is the fact that the stakeholders are allowed to interact with parts of the system at an early stage and influence the development of the requirements. On the downside, prototypes are often both time and cost consuming, and users might get attached to early prototypes which prevents them from accepting other solutions. [11]

Vijayan and Raju [16] performed a case study where they examined the use of paper prototypes for requirements elicitation. They stated that research has shown the difficulty among users to correctly explain their requirements with only using words. Users find it easier to communicate their wants and needs when they see an actual part of the system and can try it out for themselves. The case study was performed based on the following steps [16]:

- **Domain Knowledge Acquisition** - the analyst has to gain domain knowledge and understand relevant terms and processes.
3.4. Usability Evaluation

- **System Understanding** - by reading manuals and documentations and study the current user of the system the analyst will gain relevant knowledge of the existing system.

- **Requirements Elicitation** - after the previous two steps the analyst has a general idea of the system and its requirements so he/she can create a throwaway paper prototype. This prototype is presented for the user and during a discussion between the analyst and the user further requirements are gathered.

- **Paper Prototype Validation** - the requirements are categorised and the prototype is validated for omissions and ambiguity.

- **Requirements Stabilisation** - the prototype is changed according to the requirements and feedback from the user, until the user is satisfied and all the requirements are met. Once the stabilisation process is finished, the requirements specification is done and the prototype has fulfilled its purpose and can be thrown away.

The results from this case study showed that most of the participants needed 1-3 iterations to complete the elicitation process and a majority recommended this type of technique for requirements elicitation in small and medium sized projects. By using paper prototyping for gathering requirements, there are several advantages: no technical difficulties for the users since they can easily understand the system, more room for imagination and encouraging new ideas and it is user-friendly.

In the paper *Loud and Interactive Paper Prototyping in Requirements Elicitation: What is it Good for?* Abad et al. [17] conducted a study to see how interactive and Loud Paper Prototyping (LPP) could affect the requirements elicitation process in app development. The teams in the study either used the technique Silent Paper Prototyping (SPP), LPP or face-to-face meetings (without any prototype) when meeting with the client. When the teams using SPP and LPP created their low fidelity prototype, they had to use the three prototyping techniques sketching, storyboards and Wizard of Oz. Overall, the results showed that using paper prototyping for requirements elicitation was especially helpful in catching non-functional requirements in the initial state, as well as creating a more user-friendly and intuitive product. Furthermore, the teams that tended to change current requirements and add new ones to a higher extent, was the ones using LPP over SPP. A majority of the participants agreed upon LPP being the most useful technique for gathering requirements in app development.

3.4 Usability Evaluation

To evaluate a system or a product to see if it has certain usability characteristics, there exists plenty of methods to choose from. Holzinger [18] divides them into two main categories: inspection methods and test methods. Heuristic evaluation, cognitive walkthrough and action analysis all belong to the inspection methods. None of the mentioned methods is in need of any users, since the idea is instead to evaluate the system against known usability standards. The category test methods, on the other hand, needs end users and is the most fundamental usability method. Some of the most common methods in this category are questionnaires, field observation and thinking aloud, where the latter one requires a minimum of three test users.

3.4.1 Usability Testing

The term usability testing is explained by Rubin and Chisnell [19] as: "a process that employs people as testing participants who are representative of the target audience to evaluate the degree to which a product meets specific usability criteria". According to Rubin and Chisnell, the process of conducting a usability test can be divided into the following steps [19]:

- System Understanding: by reading manuals and documentations and study the current user of the system, the analyst will gain relevant knowledge of the existing system.
- Requirements Elicitation: after the previous two steps, the analyst has a general idea of the system and its requirements. So, they can create a throwaway paper prototype. This prototype is presented for the user and during a discussion between the analyst and the user, further requirements are gathered.
- Paper Prototype Validation: the requirements are categorized and the prototype is validated for omissions and ambiguity.
- Requirements Stabilisation: the prototype is changed according to the requirements and feedback from the user, until the user is satisfied and all the requirements are met. Once the stabilization process is finished, the requirements specification is done and the prototype has fulfilled its purpose and can be thrown away.

The results from this case study showed that most of the participants needed 1-3 iterations to complete the elicitation process and a majority recommended this type of technique for requirements elicitation in small and medium-sized projects. By using paper prototyping for gathering requirements, there are several advantages: no technical difficulties for the users since they can easily understand the system, more room for imagination and encouraging new ideas, and it is user-friendly.

In the paper *Loud and Interactive Paper Prototyping in Requirements Elicitation: What is it Good for?* Abad et al. [17] conducted a study to see how interactive and Loud Paper Prototyping (LPP) could affect the requirements elicitation process in app development. The teams in the study either used the technique Silent Paper Prototyping (SPP), LPP or face-to-face meetings (without any prototype) when meeting with the client. When the teams using SPP and LPP created their low fidelity prototype, they had to use the three prototyping techniques sketching, storyboards, and Wizard of Oz. Overall, the results showed that using paper prototyping for requirements elicitation was especially helpful in catching non-functional requirements in the initial state, as well as creating a more user-friendly and intuitive product. Furthermore, the teams that tended to change current requirements and add new ones to a higher extent, was the ones using LPP over SPP. A majority of the participants agreed upon LPP being the most useful technique for gathering requirements in app development.
3.4. Usability Evaluation

- **Develop the test plan** - start to create the test plan as soon as you know what you will be testing. A comprehensive test plan serves as a foundation for the whole testing process.

- **Set up a testing environment** - when choosing the location for the tests, have the following factors in mind: your test design and measures, logistics, public relations within your company and availability of participants.

- **Find and select participants** - try to identify behaviour, knowledge and characteristics of a typical user. The selected users should be as representative as possible to the actual end users of the product.

- **Prepare test materials** - test materials can for example be background questionnaires, data collection tools and task scenarios. Make sure to gather all relevant materials well in advance of the test.

- **Conduct the test sessions** - use a checklist during the session to make sure you do not miss any important steps.

- **Debrief the participant and observers** - very important to resolve any confusion or misunderstanding of why a participant acted in a specific way during the test. Debriefing with the observer can help you see the test session in other perspectives.

- **Analyse data and observations** - this step is a first and quick analysis of the worst problems encountered and then delivered to the designers.

- **Report findings and recommendations** - tightly connected to the previous step, but in this part the data is analysed in more depth and you develop findings and recommendations which result in a final report.

Both Tan, Liu and Bishu [20], as well as Maguire and Isherwood [21] have in their respective articles conducted a study comparing user testing and heuristic evaluation when identifying usability problems on websites. It is important to keep in mind the positive and negative aspects of the existing usability evaluation methods, since the choice of method could effect the outcomes of a usability evaluation. Both studies reach the conclusion of user testing and heuristic evaluation being more complimentary rather than competing, based on the differences found. They also suggests the two methods should be used at different stages in the development process, where heuristic evaluation is more suited for the initial parts of a project and user testing is best performed later on. Tan, Liu and Bishu [20] concluded that even though user testing found less usability problems, the possibility of finding problems at a deeper detail level was significantly higher. Furthermore, as user testing is believed to be the best fit later on in the development process, new usability problems might be detected as the interface design is refined and improved. Adding to this, from the results conducted by Maguire and Isherwood [21], user testing took less time to perform and found slightly more severe usability problems.

Tan, Liu and Bishu [20] also brings up the discussion regarding how many participants are enough to find an adequate number of usability problems. This has been an ongoing debate since Nielsen and Landauer [22] published their article saying that five participants were enough to find 85% of the problems. They stated that increasing the amount of participants would decrease the number of problems found by each individual participant. The findings by Nielsen and Landauer has been widely accepted and viewed as “industry standards”. However, Tan, Liu and Bishu [20] present results saying that five evaluators only could find 35% of the problems, which is the same results as Spool and Schroeder [23] presented in their article *Testing Web Sites: Five Users Is Nowhere Near Enough*.

In addition to this debate regarding the number of participants in usability testing, Lindgaard and Chattratchart [24] has published the article *Usability Testing: What Have We...*
3.4. Usability Evaluation

Overlooked? and says the focus should be shifted to task coverage instead. They present results which indicate that focusing on wide task coverage, instead of adding more users to the testing procedure, is more successful. It is better to create more tasks and give to a smaller group of end users for testing, rather than creating a few tasks and test among a larger group of end users. Similar findings has been done by Alshamari and Mayhew [25], who conducted a study where they examined the effect of using two different types of tasks during usability testing. They used structured and unstructured tasks and part of the aim was to investigate the impact task design had on discovering usability problems. The results showed that the group who received structured tasks discovered 85% of the usability problems, while the group with unstructured tasks discovered 53%. But when categorising the discovered problems according to levels of severity, both groups basically found the same amount of usability problems at the level of disaster, which was the highest level. Alshamari and Mayhew concludes their findings by stating that different tasks can discover different types of usability problems and that there is a need for more research in the area of task design since it clearly is an aspect that can influence the usability testing results.

3.4.1.1 Prototyping

In the context of usability testing, prototyping is a technique for evaluating the system or product throughout the development process. One can either use low-fidelity prototypes, which usually is made by pen and paper and can differ much from the final product, or one can use high-fidelity prototypes which are much more similar to the end product. High-fidelity prototypes are often seen as much more expensive and time-consuming to develop than low-fidelity prototypes. [26]

Walker, Takayama and Landay [26] have investigated the difference in using low- or high-fidelity prototypes as well as paper or computer as prototyping medium when performing usability testing for web applications. The aim was to determine which technique was the best, in terms of detected usability problems and cost and flexibility for the designers. No significant differences were detected among either the level of fidelity or the medium used when it came to the amount of discovered usability problems. Therefore the authors suggested choosing the technique most suitable for the specific project and current circumstances. Although low-fidelity prototyping was seen as less time-consuming and required a lower level of knowledge by the designer, it could also have some limitations regarding the range of interaction techniques. The users tended to leave more comments on the computer-based prototypes compared to the paper prototypes, but it had no influence on the number of discovered usability problems.

Other studies regarding different types of prototyping techniques have been conducted, for example in the article The Use of Paper-Prototyping in a Low-Fidelity Usability Study written by Olmsted-Hawala, Romano and Murphy [27]. They conclude that the use of low-fidelity paper prototyping is successful at early stages in the development to detect design issues and problems. Even though the user interface might lack some sense of reality, other benefits are believed to overcome such drawback. For example the easiness of implementing changes since the medium is paper, or the fact that users appeared to be more willing to express their own ideas and possible improvements on the design when the interface was presented on a piece of paper. Still and Morris [28] also investigated in the paper prototyping technique, but with a different approach by adding the blank-page technique. This technique was used to get a better understanding of the users thoughts and ideas when testing the interface. If a user entered a part of the interface that was not yet implemented, a short text message was presented and encouraged the user to either write or draw what he or she wanted to see on that blank page. The authors conclude that this type of technique makes usability testing more effective and forces the users to actually think more about their own expectations regarding the system and sharing more ideas for potential improvements. Moreover, the authors state
that paper prototyping is a technique suitable for the later stages of development, and not only for the earlier stages.

### 3.4.2 Thinking Aloud

Thinking aloud is a technique where the user is asked to verbalise his or her thoughts while exploring the interface of a system. This greatly increases the evaluators understanding of how the user views the system and what might cause certain misconceptions. Advantages with the thinking aloud technique is the retrieved knowledge of why users act in a certain way and how the user actually uses the system. It can also help some users to focus on the task and as a usability tester you can retrieve a great amount of data by using only a few participants. However, when users are very focused it can also be a drawback since it might decrease the otherwise natural interactions.

In all the studies some form of usability testing is performed and the user is encouraged to think out loud while exploring the user interface. This could indicate that combining user testing and thinking aloud is a useful approach. According to McDonald, Edwards and Zhao and their study, 98% of their respondents said they had used the thinking aloud method sometimes and 71% saying they use the method on a regular basis. Most commonly was the use of the method in usability studies, where 86% answered they would use it in early stages of the process and 67% said they would use the method in the end of the development process.

### 3.4.3 Validity and Reliability

Regardless of conducting a quantitative or qualitative study you need to think about validity and reliability. This also concern usability evaluation studies. Validity can easily be described as making sure you are really measuring what you are supposed to be measuring and reliability is the concern regarding whether the same results can be reached once again when measuring, even if some of the conditions might differ. The aim is to reach high validity and reliability.

In the article *Understanding Reliability and Validity in Qualitative Research* by Nahid Golafshani one way to maximise the validity and reliability of a qualitative study is said to be by using triangulation methods. By using several kinds of data or combining different methods, both quantitative and qualitative approaches, it could strengthen a study. But the methods chosen in a triangulation approach to test the validity and reliability of a study is not pre-determined. It depends on the study and its criterions.

When it comes to validity, there are several different types and they are mainly divided into two categories: instrument validity and experimental validity. Instrument validity regards the instruments or measures used in the study and has the three subtypes construct validity, content validity and face validity. Experimental validity is related to the possibility of generalising the results and also has three subtypes which are internal validity, external validity and ecological validity.

### 3.5 Errors

Norman categorises errors as either slips or mistakes. A slip is when the user intends to do one action, but ends up doing something else. The user had a correct goal in mind, but the performed actions did not fulfill the goal. Whereas a mistake is when the goal itself is wrong and therefore the following actions are also wrong. Even though they might be correct for completing that specific goal. Slips and mistakes can be related to the steps in the seven-stage of action cycle presented by Norman, which also can be used to understand human behaviour.
3.5. Errors

- **Goal** - form the goal.
- **Plan** - the action.
- **Specify** - an action sequence.
- **Perform** - the action sequence.
- **Perceive** - the state of the world.
- **Interpret** - the perception.
- **Compare** - the outcome with the goal.

Slips occur either in performing the plan or when perceiving or interpreting the results, while mistakes usually happens when setting the goal, making the plan or comparing the results with the goal.

In the article *Usability: A Critical Analysis and a Taxonomy* by Alonso-Ríos et al. [8] they create a new taxonomy for usability attributes based on previous research and definitions. One of the proposed attributes is robustness which is defined as: “the capacity of the system to resist error and adverse situations”. It is said to be closely related to two other attributes defined in literature: Errors as defined by Nielsen [3]: “the system should have a low error rate, so that users make few errors during the use of the system, and so that if they do make errors they can easily recover from them. Further, catastrophic errors must not occur”, as well as error tolerance as described by Quesenbery [32]: “an error tolerant program is designed to prevent errors caused by the user’s interaction, and to help the user in recovering from any errors that do occur”.

Nielsen [3] states that errors should be separated depending on their impact. Either an error occurs which the user immediately notices, can quickly correct and only affects the user as in the transaction time it takes to complete the current task. The other type of error might pass the user unnoticed and lead to catastrophic results such as destroying the user’s work. To measure the error rate during usability testing one can either count the errors over a fixed period of time, or while the user performs a certain number of tasks [33]. Another perspective is to measure the impact the errors have, such as how long it takes for the user to recover from an error, or how much the error costs in money for the targeted business.

3.5.1 Error Prevention

There are many reasons why errors occur, but according to Norman [4] the most common reason is because users are put in environments where they are expected to act in unnatural ways. For example staying focused for many hours at a time, multitasking and handling several interfering actions. Another common reason for errors to occur is because it is difficult to resume an action once an interruption has occurred. Quesenbery [32] adds to this discussion by saying that errors could also occur because the designers did not cover all possible ways a user could interact with the system.

After knowing why errors occur, you also want to know how you can prevent them. Nielsen [3] says for example that every situation when the user is asked to spell something, spelling errors can occur, and therefore a better solution would be to provide the user with predefined input to choose from, if appropriate. Other actions, which might have more serious consequences if not ruled out correctly, could ask the user for confirmation before executing. Lastly Nielsen mentions modes which are a common source of user errors. A mode error occurs when a user tries to perform an action not belonging to the current mode of the interface. One simple example is the Caps Lock key, which has two different modes where the user either can type uppercase or lowercase letters. Nielsen suggests designs to clearly

---

state which mode is currently active and to differentiate between different modes so as to avoid errors occurring.

From the article “What Does Usability Mean: Looking Beyond “Ease of Use”” [32] Quesenbery derives some design guidelines for preventing errors:

- Make it difficult to take incorrect actions.
- Make it difficult to take invalid actions.
- Make it difficult to take irreversible actions.
- Plan for the unexpected.

A more concrete technique, rather than examples and guidelines, is the Technique for Human Error Assessment (THEA) developed by Pocock, Harrison, Wright and Johnson [34] with the aim of helping designers and engineers to forecast interaction failures. The technique is based on a model of human information processing, but created to be easy to use and understand for those who do not have deep knowledge of human factors or cognitive psychology. By using scenarios the technique focuses on what actually happens in reality, rather than what designers think will happen. The important step in THEA is the error analysis, which consists of created statements related to the derived scenarios and categorised in one of the following categories: Goals, Plans, Performing actions or Perception/Evaluation/Interpretation. These statements can either be true or false and are intended to reveal where in the design failures might occur and their impact.

From THEA and Normans seven-stage of action cycle, mentioned in 3.5, a design principle is derived, called anchor-based subgoaling (ABS). This was used in the article “User Interface Dependability through Goal-Error Prevention” [2] written by Reeder and Maxion. The aim was to reduce goal errors (errors that happens when users do not understand what to do) when changing file permissions in the existing Windows XP interface, by implementing another interface derived from the design principles. Anchor-based subgoaling is a principle supposed to make sure that an interface contains the right information based on the task the user wants to complete. Furthermore, it ensures that the provided information is comprehensible and displayed so the user can easily notice.

Before the process of ABS can start, one have to perform task analysis. A good alternative is the hierarchical task analysis, presented by Kirwan [35], which divides a task into a series of subgoals and relevant operations necessary to achieve them. Each task has one primary goal as the root. [2]

The ABS design procedure then contains the following steps [2]:

- **Phase 1** - Identify the information that is needed.
  1. For all the goals created in the task analysis, do the following:
     (a) identify all the information a user would need to determine that a goal is completed.
     (b) identify all the information a user would need to arrange the subgoals beneath the primary goal.
  2. For all the operations in the task analysis, identify what information a user would need to perform the operation.

- **Phase 2** - Provide the information in the interface.

Based on the results from the study, Reeder and Maxion [2] can conclude that by following the anchor-based subgoaling design principle when creating the new interface, they succeeded in reducing the goal errors and improved the task-completion for setting file permissions. They further state that by presenting comprehensive information related to the users primary
goal, the amount of goal errors can be decreased. However, they do declare that the technique has to be tested together with other types of tasks before it can be widely accepted.

But if errors still occur (which they probably will do to some extent) one approach to follow to create good error messages is the one denoted by Nielsen [3]:

- They should be phrased in clear language and avoid obscure codes.
- They should be precise rather than vague or general.
- They should constructively help the user solve the problem.
- They should be polite and not intimidate the user or put the blame explicitly on the user.
This chapter presents the method used in this study, carefully described with as much details as possible to ease the process of someone else wanting to reinterpret the study. All choices taken during the process are motivated and explained.

4.1 Structure of the Study

The study was mainly divided into two bigger parts, first the pre-study and second the implementation together with the evaluation. A pre-study is neccessary to get as much information and knowledge as possible before doing the actual work. The pre-study included both a literature study and the gathering of requirements. According to Nielsen [3], in usability engineering it is common to work with an iterative process. One reason for this is because solving one usability problem might introduce new ones. Therefore, the work after the pre-study was planned in an iterative manner with two iterations. An iteration included both the design phase of a prototype and the evaluation with user testing.

4.2 Pre-study

The pre-study started of with a literature study to gather information regarding usability engineering and specific methods to use when evaluating interfaces, how to gather requirements during a process, and when human make errors during human-computer interaction and how to prevent them from happening. Information was also gathered regarding flight tests and the old program, which is currently used during flight test. When searching for literature Google Scholar was mostly used and to some extent the databases IEEE, ACM and Springer. The aim was to use well-cited scientific papers which had been published recently. The chosen papers were also used further in the search for relevant resources by looking through the reference list, as well as the Google Scholar function “Citations to this Article”. When struggling to find papers meeting both of the requirements (well-cited and recently published), papers were accepted which met one of the requirements. Books were also used to complement when not enough information could be found in the scientific papers.
4.2 Requirements Elicitation

In this study, two types of techniques were used in the requirements elicitation process: interviews and prototyping. Interviews were chosen since it is a commonly used technique and it provides the researcher with a lot of data in a quick manner [11]. Prototyping was chosen as the second technique because it is often used together with interviews [11] and because users often find it easier to communicate their wants and needs when they are provided with a prototype [16]. These techniques were combined and used during the sessions for gathering requirements.

Participants

For the requirements elicitation process three participants were chosen and presented in Table 4.1. The participants had to have knowledge in the field to make sure the right requirements were gathered. Two of the participants had worked with the old program and will also continue to work with the new one, while one of the participants had not used the old program. This third person was included to get unbiased thoughts and ideas that might be of value and this person had knowledge in the field and worked with tasks closely related to the old program. The number of participants in the requirements elicitation process was mostly based on the availability of people with relevant and enough domain knowledge.

<table>
<thead>
<tr>
<th>Participant</th>
<th>Title</th>
<th>Domain Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Engineer</td>
<td>10 years</td>
</tr>
<tr>
<td>2</td>
<td>Engineer</td>
<td>20 years</td>
</tr>
<tr>
<td>3</td>
<td>Engineer</td>
<td>38 years</td>
</tr>
</tbody>
</table>

Interviews

First of, a plan for the structure of the interview session was created. It can be found in Appendix A. The interview questions, which can be found in Appendix B, were mainly formulated based on the knowledge retrieved during the pre-study, but also inspired by the study Bano et al [15] conducted by trying to avoid the mistakes most novices make when performing interviews. For example, each question was carefully formulated to minimise misunderstandings and trying to make each question as clear as possible. Moreover, the order of the questions were considered and strived towards having more general questions at the beginning and ending with more detailed questions. The aim was to create open questions, meaning not just yes- or no-questions, to increase the possibility of retrieving a greater amount of information.

The interviews were performed with one participant at a time and with a semi-structured approach, to make it possible to delve into new thoughts and ideas. All the interview sessions were conducted in the same environment. To make it easier for the interviewer to maintain focus, all the interviews were recorded after agreement with the participant. The interviews started with general information regarding the participants right to cancel the session at any time and that it was voluntary to take part, the fact that every participant was anonymous and that the results were only going to be used in this specific study. After the interview questions had been asked, the session continued by showing the prototype.

Prototyping

The prototype was created with pen and paper and mainly used for the requirements elicitation process. The aim was to create a first draft of how the interface could look like, but with
a small amount of details. From the start, a few requirements had already been stated by the client along with the description of the program. These requirements were the inspiration for this prototype and they were formulated as follows:

1. Enter the position and attitude for the trackers and the slaves. These should be saved and able to reuse.
2. Handle up to 6 different trackers.
3. Each tracker should be able to connect with several slaves.
4. Connect the trackers and the slaves.
5. Start and stop the program.
6. Replay a conducted session.
7. Save events from the program in an event log.
8. Save all input and output to/from the program in a track log.

The prototype was used after the interview questions had been asked and the participants were informed with the purpose of the prototype, to serve as another tool for gathering requirements. Furthermore, they were told that it was a first draft and not necessarily an indication of how the real interface was going to be implemented. The prototype was shown to the participants who were informed to investigate it and encouraged to share all sorts of thoughts and ideas, both positive and negative. No specific questions were formulated beforehand regarding the prototype. The idea was to have a discussion with the participants based on their shared thoughts and opinions.

4.3 Implementation and Evaluation

The next step was the implementation and evaluation, which was done with an iterative approach. Two iterations were conducted and the main reason behind the number of iterations was the time aspect. The implementation phase consisted of designing and creating a prototype, which was then used in the evaluation phase. During the evaluation, user testing together with the thinking aloud technique was used. The choice of user testing as the evaluation technique was based on the fact that it does not require a lot of time to perform and does not require any experts, but preferably users with some domain knowledge. Furthermore, the combination with the thinking aloud technique was believed to reveal more ideas and get a deeper understanding of the users interaction with the program. According to several studies, as mentioned in Section 3.4.2, a combination of user testing and some form of the thinking aloud technique is a common approach.

Test plan

Before starting with the first implementation a test plan was created, which you should do as soon as possible according to Rubin and Chisnell [19]. The test plan was used as a foundation for the testing process and to make sure each test session differed as little as possible from the previous ones. It included for example steps regarding the preparation of each test, a checklist to use during the test session as well as important actions to take after each session was done.
Participants
For the user testing process, three participants were used in the first iteration and four participants were used in the second iteration. The participants were chosen based on their domain knowledge and the fact that they had worked with the old program or would work with the new program, or with tasks closely related and highly relevant. The number of participants were mainly based on the available persons with relevant domain knowledge, and according to Holzinger [18] the thinking aloud technique requires a minimum of three test users. Regarding user testing, there is an ongoing debate about how many participants are necessary to find enough usability problems. Adding to this debate, the researchers Lindgaard and Chattratichart [24] as well as Alshamari and Mayhew [25] tries to shift focus from the number of test users to the importance of creating good enough tasks used during the user tests instead. This study was inspired by these researchers in the means of focusing more on developing good tasks rather than the number of participants in the user testing.

4.3.1 Implementation: Low-fidelity Prototype - Iteration 1
Based on the outcomes from the interviews and the prototyping, together with the initial requirements already listed, a first prototype of the actual interface was created. The design phase was also inspired by the ABS design procedure, described in section 3.5.1. By identifying the goals and purposes the participants would have with using the program, and together with the requirements, trying to determine what information is needed to meet these goals and purposes. This information was then used as guidelines when deciding on how to design the interface.

The low-fidelity prototype was developed by using Microsoft PowerPoint. Each slide represented a view in the interface and contained figures and text boxes to present necessary information and allow for user interactions. All pages were printed before being used during the user tests. The reason behind the use of a computer program instead of pen and paper when creating the prototype was the ease of making changes quickly. Another idea was that the participants, during the user tests, might not focus too much on minor layout differences, such as the size of buttons, text boxes or font size, if the whole interface was created by using a computer program. But, at this stage in the development it was still supposed to be a low-fidelity prototype, so therefore the pages were printed and used during the user tests, instead of displayed on the computer-screen.

4.3.2 Evaluation: User Testing - Iteration 1
The evaluation was performed as three separate test sessions and the participants who attended the tests are presented in Table 4.2. Each session started with general information regarding the test users voluntary participation, their right to cancel and leave the session at any time, followed by the aim and the structure of the session. After agreement with the user, each session was recorded to minimise the need of taking notes. Although, some notes were taken to highlight specific interesting sections of the session and remembering them when listening to the recording later on. Especially situations were the user had trouble understanding the prototype, made mistakes or when errors occurred.

Table 4.2: List of participants in the user tests in iteration 1.

<table>
<thead>
<tr>
<th>User</th>
<th>Title</th>
<th>Domain Experience</th>
<th>Involved earlier</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Engineer</td>
<td>10 years</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Engineer</td>
<td>20 years</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Engineer</td>
<td>10 years</td>
<td>Yes</td>
</tr>
</tbody>
</table>
The user was presented with the prototype and told to solve several tasks, which are presented down below. The tasks were created based on the initial requirements and the outcomes from the interviews and prototyping in the requirements elicitation process. One task was given to the user at a time and the user was encouraged to think aloud as in explaining the choices taken as well as share any kind of thoughts and ideas. Since the prototype was created on paper, the response to the user’s interaction with the prototype had to be illustrated by manually providing the user with a new view of the prototype. The prototyping was also inspired by Still and Morris [28] who used a blank-page technique. If the user ever entered a part of the interface which was not yet implemented, a short message was presented and told the user to either write or draw how he or she wanted that part of the interface to look like.

While the user examined the prototype and performed the given tasks, the test moderator took notes every time the user made a mistake. Mistakes included both minor things such as misspelling or mistyping, as well as wrong actions that did not lead to reaching the intended goal of the task. An action was classified as a mistake even if the user detected and corrected it by him- or herself.

The following tasks were presented to the user during the test session in iteration 1 and the descriptions here are to some extent shortened since the whole descriptions can be found in Appendix D:

1. Create a new session in a sequential/parallel view. Add a tracker and a slave (with predefined data). Connect the tracker and the slave.
2. Change the position of the slave.
3. Start the session. Mark that event 3 happened. Stop the session.
5. Replay a test session. Close the replay and exit the program.

Task number 1 and number 4 was either telling the user to create a new session in a sequential view or in a parallel view. If task number 1 had the sequential view then task number 4 had the parallel view, and vice versa. The idea was that every user should experience both views. The order of task number 1 and task number 4 was also shifted between the user tests. If the first user was presented with task number 1 as the first task, then the next user got task number 4 as the first one instead. Both tasks were about creating a new test session, but with a slightly different approach. The idea was to examine if the users behaved differently depending on the slight difference in the task description. All the other tasks presented to the users had the same order in all the sessions.

When the user had performed all the provided tasks, the session continued with the user being asked several questions, as presented down below. The idea was to gather even more information and reveal any misunderstandings regarding the prototype or the tasks. At this stage, more of a discussion between the user and the test moderator was encouraged. While earlier in the session, the test moderator had a role of observing rather than guiding or discussing with the user.

The following questions were asked after the user had performed all tasks:

1. What was good with this prototype?
2. What could be improved with this prototype?
3. What was difficult regarding task number 1 - 5?
4.3.3 Implementation: High-fidelity Prototype - Iteration 2

By using the results from the user testing in iteration 1, the prototype was modified and a high-fidelity prototype was created. This prototype was created by using Tkinter, the standard GUI library for Python. The initial requirements and the results from the interviews during the requirements elicitation were considered when refining the prototype. Basically to make sure that new changes still aligned with the aim of the program. Where the results from the user testing indicated that users tended to do mistakes or take actions that ended in errors, the guidelines formed by Nielsen were considered to design good error messages.

The idea was to create a prototype representing all the functionality needed in the program and for the prototype to be as alike the final interface as possible.

4.3.4 Evaluation: User Testing - Iteration 2

The evaluation was performed as described in section 4.3.2, but this time with four separate test sessions. The participants who attended the tests are presented in Table 4.3. Since the prototype used during this evaluation phase was created and displayed on a computer, the screen was also recorded by using the built-in recording function in Microsoft PowerPoint.

<table>
<thead>
<tr>
<th>User</th>
<th>Title</th>
<th>Domain Experience</th>
<th>Involved earlier</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Engineer</td>
<td>20 years</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Engineer</td>
<td>10 years</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Engineer</td>
<td>38 years</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Engineer</td>
<td>10 years</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 4.3: List of participants in the user tests in iteration 2.

The following tasks were presented to the user during the test session in iteration 2 and the descriptions here are to some extent shortened since the whole descriptions can be found in Appendix D:

1. Create a new setup with one tracker and one slave (with predefined data). Connect the tracker and the slave. Change the height of the slave and save the setup.
2. Perform changes to a setup-file by changing the position of a tracker. Save the setup-file.
3. Create a new session and load a setup-file. Create and name three buttons for event-registration. Start the session. Mark that any of the three events happened and then stop the session.
4. Replay a session by choosing a track log. Start the replay. Wait for the replayed session to end and then exit the whole program.

The tasks were created based on the ones from the user testing in iteration 1, but with some minor changes. When the user had performed all the provided tasks, the session continued with the user being asked the following questions:

1. What was your overall impression of the prototype?
2. How easy was it to make a mistake in task number 1 - 4?
This chapter presents the results, both from the pre-study and the implementation and evaluation of the user interface. The sections are structured in the same order as the method chapter to make it easy to follow the process and the outcomes.

5.1 Pre-study

The main part of the pre-study contained the process of gathering requirements, thus the results presented in this section is based on the outcomes from the requirements elicitation.

5.1.1 Requirements Elicitation

In Appendix B the interview questions are presented and in Appendix C the paper prototype for the requirements elicitation process is shown. The results from this session is a summary of what each participant said during the interview, including comments and thoughts shared when being presented with the prototype. A list of requirements is also presented, created based on the outcomes from the sessions with the participants.

Participant 1

The first participant had not used the current program, but was working with tasks closely related to the use of the program. Therefore the answers given during the interview had a more hypothetical touch, since the participant did not have actual experience with the program. The participant believed that the program was used at every flight test as well as ground level test, and that the purpose was to align the slaves based on the information retrieved from the trackers. One important task was to make sure that the different slaves had the right positions and the correct angles (azimuth and elevation) to point to the same object as the tracker. If the tracker and the slave would not be pointing to the same object, the participant said the program would be useless. Therefore it would be desirable to in some way verify, before a flight test has started, that the two systems are pointing to the same object.

Regarding the paper prototype, one suggestion the participant had was to save information regarding the specific test session (purpose of the test, participants etc) in a file which could then be presented somewhere in the interface. Another suggestion was to have the op-
portunity to leave comments in the program related to events happening in real time during a flight test, and to connect the program to another program called SIMDIS. SIMDIS can in real time present the trackers and the slaves positions and angles visually, and show if the two systems are pointing to the same position. The comments made by the user should then be saved in the event log together with a time stamp, and if the session is later replayed, the program should notify the user everytime it reaches such a comment. Lastly, the participant suggested to keep the interface as simple as possible, use more colours as well as symbols to represent different actions such as starting and stopping the program.

Participant 2

The second participant had used the old program one time before and is working in the same domain as the program will be used in. The participant said that the program is used during flight tests with sensors that needs to be aligned with a flying object and the main purpose of the program is to send correct data from one system to the other, so they can point in the same direction (usually towards a flying object). Up until now, the program has only been used a handful of times but the participant believed it would be used more frequently in the near future. Especially if the functionality can be extended. The most important task in the program would be its core function, meaning the task of sending correct data from the tracker to the slave (placed at different locations) and making sure they point to the same object at the same time. Apart from that, the participant also wanted presentation of real time data, being able to register and log all relevant data during a test session, and use the program with the same functionality but for ships instead of flying objects. When being asked to prioritise the different alternatives mentioned, the participant found it difficult to do so. Regarding the old program’s interface, the positive aspects was the fact that you could add several systems (both trackers and slaves) at the same time. One tracker could for example have three slaves while another tracker had one slave, and they were all used at the same time during a flight test. The participant did not have any specific negative comments, apart from the lack of presenting real time data during a test session.

When being presented with the paper prototype, several ideas and suggestion emerged. A first suggestion was to split the information on the first view into two, where one view had the functionality of adding the actual trackers and slaves and their positions. In the other view you could have the opportunity to connect the different trackers and slaves, for example one tracker with two slaves. Regarding the desire of using the program together with ships instead of planes, a separate view could be designed with functionality related to that specific case.

The task of adding and connecting the trackers and slaves should be summarised in a setup-file, which later could be chosen from a list of other setup-files and used when starting the actual test session. It should be clear which setup is chosen and what type of information that has been added (position and angles of the systems). When the test session is up and running, all data that is being sent to/from the program should be documented in a Track Log. This log should contain information such as time and date, which setup was used etc. Somewhere in the interface you should also be able to see all the available track logs, search for a specific one by name and export them to CSV-format.

The participant had a suggestion of not only displaying a start and stop button, but also a pause button. This button could be used if the flying object is supposed to disappear for a while, and then come back again. During this period no data should be logged in the Track Log, since the flying object is not present. The idea is that the data logged before and after the pause should appear in the same file, related to the current test session.

When the session is up and running and the flight test has started, the participant described another idea of having some sort of indication of the stream of data to/from the

\footnote{https://simdis.nrl.navy.mil/}
5.1. Pre-study

program. A lightbulb blinking every time data was sent from a tracker to the program, and another one everytime the program sent data to a slave. The lightbulb could also have a constant light instead, as long as data is being received, and if no data is received within a certain time interval it could be turned off.

As mentioned earlier, the participant wanted to have a view showing the flight test in real time, and preferably with a connection to the program SIMDIS which could then serve as the real time view. The purpose would be to determine that everything is working during a test. For example, to illustrate that the tracker actually have found the flying object, a symbol can be placed in the real time view and have a specific colour. If the tracker loses track of the target, the symbol representing the target can change to another colour. This connection with SIMDIS would also be valuable to have when wanting to replay a conducted session.

Lastly, the event log and related functionality was discussed. The participant wanted both system events (such as errors when executing the code) to be logged in the event log, but also the possibility to save events added by the user when the program and test session is up and running. Different types of events could be predefined and related to different keyboard shortcuts, and the meaning behind these shortcuts could be presented somewhere in the interface. Either could these system events and user added events be stored in the same event log, or in separate ones. The event log should also be exportable.

To get a more coherent view, the participant suggested that the very first step in the program could be to name the test session and then both the track log and the event log would have that same name, but with different endings to separate them from each other. The filenames should be consistent even after exporting the files.

Participant 3

The third and last participant had used the current program five-six times over the years, and had the most experience with the program. The participant said that the program is used to connect different systems and direct them to point to the exact same position. It is a very useful program to have and an important part during a flight test. One important aspect of the program is for it to be user-friendly, and another one is for it to be as generic as possible so it can connect to several different types of trackers and slaves. Furthermore, the participant wanted the program to have some sort of prediction, such as a Kalman-filter, to predict the path of the target. One challenge the participant could see with the program was how to design the user interface to make it understandable and easy to use. As for today, the interface has too many restrictions which almost demands for a daily use. The participant would rather have a sequential design when setting up a test session, to minimise the risk of doing a mistake or forgetting something, since a flight test is very expensive to perform. One suggestion the participant had was to design two separate views, one which had a more sequential approach and other one which had all the information and options displayed at the same time. The user could then choose between these views depending on how much the person remembered from last time the program was used. A user manual was also said to be mandatory.

When the discussion moved over to the paper prototype, one suggestion the participant had was a function to test the connection between the trackers and the slaves before the test session was started. The program could either ping the connected slaves, or send out a predefined function to see if the slaves sensors were moving. As of today, this type of test is not done at all. Another idea was to display the IP-addresses of the respective slaves which was a part of the current test session. Furthermore, the participant would like to add a start and a stop button related to the data stream which comes from the tracker. Even though the test session has not started and the tracker has not discovered the target, data streams could still be sent to see that the whole system is working. The other start and stop button can then be used to indicate that the test session is actually started. By clicking that start button a
5.1. Pre-study

time stamp should be added in the track log to indicate that from now on the recorded data belongs to the test session.

This data stream, coming from the tracker, should be saved in a track log and the participant also described an idea with a button or symbol that would indicate that the tracker had found the target and was now following it. The track log should contain data from the trackers, which serve as input to the program, and data sent to the slaves, which serve as output from of the program. Regarding the event log, this participant also wanted to be able to mark and save events happening during a test session. To mark an event could either be done with keyboard shortcuts or with a mouse click. However the participant wanted to have some sort of response of what type of event it had chosen. After the test session the user should be able to edit the event and leave a more detailed comment. The event log should be divided into two sections, one regarding the system and code errors and the other one regarding the events entered by the user during the test session.

The participant also thought that a solution with a connection to SIMDIS to replay a conducted test session would be a very good idea. One suggestion was to have the possibility to either choose a view where you could see the whole test session and the trackers and the slaves moving, or to only see a specific snapshot from the test session.

Requirements

Based on the outcomes from the three interviews and prototyping sessions, a list of requirements for the user interface was created. The initial requirements for the program, mentioned in Section 4.2.1 were considered as well. Down below is the final list of requirements for the user interface.

1. Register trackers and slaves by entering positions and attitudes.
2. Register up to 6 trackers during one test session.
3. Connect trackers and slaves. One tracker should be able to connect to more than one slave.
4. Load a setup-file (containing position and attitudes for trackers and slaves as well as their connection which has been saved from an earlier session) into a new session.
5. Start, stop and pause a test session.
6. Visual indication for both receiving data from the tracker(s) as well as sending data to the slave(s).
7. Visual indication when the tracker(s) has/have found the target and is following.
8. Keyboard shortcuts or buttons displayed in the interface connected to predefined events, which the user can enter during a test session (and saved in an event log).
9. Have functionality for connecting an AIS receiver as a tracker.
10. Connect the program to SIMDIS for real time view and for replaying a test session.
11. Load a track log (containing data from the tracker(s) and data sent to the slave(s) which has been saved during a test session) to replay a test session.
5.2 Implementation and Evaluation

This section will present the results from the implementation and the evaluation from both iterations. From the implementation phase two prototypes emerged, a low-fidelity and a high-fidelity. The evaluation phase consisted of user tests and the outcomes from these sessions were both quantitative and qualitative data.

5.2.1 Implementation: Low-fidelity Prototype - Iteration 1

The low-fidelity prototype consisted of thirteen pages and can all be found in Appendix E. Some of the pages will also be presented here, like in Figure 5.2.1 which is the start page. From this page the user can either click the manual button to retrieve the user manual for the program, or click the start button which reveals three options: create a new test session, replay a test session or quit the program.

If the user chooses to create a new test session it can either be done in a sequential view, meaning by step-wise instructions, or in a parallel view where all the needed data can be entered in the same view. Figure 5.2.1 shows the parallel view when creating a new test session. The idea behind having a sequential view was to reduce the number of mistakes made since each step in the creation of a new test session is mandatory. This view is especially useful if the user has not used the program in a while. The parallel view was added as an alternative if the user recently used the program and is more familiar with the process of creating a new test session.

As seen in Figure 5.2.1 the user can either choose to load a setup or create a new setup. A setup consists of one or several trackers and slaves, their positions and attitudes as well as their IP-address and port number. It also contains the connections between the trackers and slaves, meaning which tracker is sending data to which slave. The idea is to save each setup from each test session and make it possible to reuse them.

In the same figure, there are two buttons saying choose tracker and choose slave. When the user clicks these buttons a list of available trackers and a list of available slaves is shown.

At the bottom of the same figure are four boxes. They are used to connect the added trackers and slaves so the program knows which tracker is sending data to which slave.

Figure 5.2.1 is the page which is presented to the user when the start button has been pushed. The window showing the Earth is an illustration of the program called SIMDIS. The idea is that SIMDIS will start as soon as the test session is started and in SIMDIS the
trackers and slaves will be visually presented live. Below the SIMDIS illustration are the three buttons for starting, stopping and pausing the test session. To the right are several shortcuts represented as buttons followed by a description. These buttons are meant for the user to click during the test session to mark that specific events occur, when watching the live presentation of the trackers and the slaves in SIMDIS. The events will be saved in a log which the user can open after the test session.

The rest of the pages for the low-fidelity prototype can be seen in Appendix E and illustrates the following: the steps taken in the sequential view when creating a new test session, a summary of the added information for the trackers and the slaves and the option for the user to do changes, a blank page which is reached when the test session is stopped and requires the user to use the start button to move forward, and also the page for replaying a test session which basically only provides a new window with the SIMDIS program. From all the pages the user can reach both the start button and the manual button.
5.2. Implementation and Evaluation

5.2.2 Evaluation: User Testing - Iteration 1

The results from the user testing is a table of number of errors the users made for each task, a table over the success rate for each task and also a summary of the answers to the questions asked and the noticed behaviour of the users.

In Table 5.1 the total number of times an error occurred are presented. In this context the following types of actions were counted as errors: pushing the wrong type of button, trying to interact with parts of the interface which was not made for interaction and moving to a page which was not relevant for solving the specific task.

Table 5.1: Total number of times an error occurred during user testing in iteration 1.

<table>
<thead>
<tr>
<th></th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of errors</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

In Table 5.2 the success rate for each task can be seen. Each X represent the outcome of one user performing one task. A task was seen as successfully completed if the user was straight forward and only performed the neccessary actions for the task and the test moderator did not have to provide any guidance. If the test moderator noticed uncertainty from the user and had to ask questions such as "What do you want to do from here?" or "How would you perform that action?" to get an action from the user, the task was marked as successfully completed with minor guidance if the guidance resulted in the user reaching the end of the task. To mark a task as partially completed the user did not follow the steps in the task description correctly, but ended up with similar results as if the user would have followed all the steps in the task description. At the bottom of the table the success rate is presented, which was calculated by only dividing the number of successfully completed tasks by the total number of users and multiplying by 100.

Table 5.2: Success rate for each task during user testing in iteration 1.

<table>
<thead>
<tr>
<th></th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Successfully completed task</td>
<td>X</td>
<td>X</td>
<td>XX</td>
<td>XX</td>
<td>X</td>
</tr>
<tr>
<td>Successfully completed task with minor guidance</td>
<td>XX</td>
<td>X</td>
<td></td>
<td>XX</td>
<td></td>
</tr>
<tr>
<td>Partially completed task</td>
<td>XX</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not completed task due to time constraint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not completed task</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Success rate</strong></td>
<td>33,3%</td>
<td>33,3%</td>
<td>66,6%</td>
<td>66,6%</td>
<td>33,3%</td>
</tr>
</tbody>
</table>

User 1

In general the user thought it was easy to understand how to interact with the interface without a user manual. However, the most confusing part was at the page where the user was told to choose which directory to save the session in. Because the user thought the text field could be used for typing the exact path to a directory and therefore did not understand why the button that followed said search. It would have been better to name that button choose and have another button for search if the user wanted to find a directory in the file explorer instead. The user also did have minor problems with finding the way to exit the program and did not think it was so intuitive to use the start button to find the exit. Instead, it would have been easier to just have a simple button to exit. The last comment the user had about the interface regarded the view where you could edit and delete the added trackers and slaves. The symbols for editing and deleting a specific tracker or slave was difficult to understand and
the user suggested either having text instead of symbols or showing a text when hovering over the buttons. Regarding the tasks given, the user said to not experience any difficulties when solving them. Although minor confusion was noticed when the user had to choose between parallel and sequential view, choose between creating a new setup or load a setup and when finding the path to exit the program.

User 2

The user thought that the starting page with the start and manual button was a good first view, but did not really understand the need for a sequential and a parallel view and especially not the naming. The main part of the comments for improvement regarded the setup-file. Firstly, the user wanted an option to name the setup-file. In this interface the user could only name the test session. When having entered the trackers and slaves and made the connections in between, all belonging to the same setup, the user wanted a save button on the same page and not on the following one. The user did not think it was necessary to move to the next page and get a summary of all the entered data and then have the save button (when using the parallel view). Another suggestion for improvement was to have a separate view where the user could handle the setup, such as creating a new one with trackers and slaves or doing changes to an existing one. For example in the start menu, so when creating a new test session the user would only need to load a setup. Moreover, the user had trouble understanding the symbols for deleting and editing trackers and slaves, just as user 1 had. Other situations where the user seemed to be confused when interacting with the interface was when trying to exit the program, when being told to save the test session since the user thought that meant you had to run a test session first and lastly when the user marked that a specific event happened and the border of the button changed and became thicker. The heading over the buttons said shortcuts which made the user believe it was not actual buttons but information of which key to push on the keyboard. Regarding the tasks, the user did not say to experience any difficulties in understanding them or performing them.

User 3

Overall the user thought it was easy to understand how to interact with the interface and it was pretty intuitive. The main aspect of improvement regarded the summary of the entered trackers and slaves. The user thought it would be a good idea to present the trackers and slaves on a map, before saving and starting a test session, to make sure the objects had the right positions. This could be done by using SIMDIS since it has such functionality. Another suggestion was making the shortcuts for events happening during a test session more user-defined. The user had an idea of creating a file for defined events which could be loaded depending on which user that used the program. Also, when marking an event the user wanted to have a popup with the possibility to enter some more text regarding the event. Moreover the user did not really understand the need for both a sequential and a parallel view, thought it was so many keystrokes for the sequential view, thought it sometimes was a bit unclear if the entered information was saved or not when moving to the next page by pushing the green arrow and wanted to have more opportunities to change entered information or revert an action. The user also had trouble understanding the symbols for deleting and editing the trackers and the slaves, just as user 1 and user 2 had in previous test sessions. None of the tasks were difficult to perform according to the user but some could instead have included more subtasks. During the user test some confusion was noticed when the user was told to create a new test session in parallel view, since the user thought that meant to open an already created test session. The user also seemed to be confused over how to save the entered trackers and slaves and how to start a test session, since no information was displayed regarding which test session that was loaded at the time.
5.2.3 Implementation: High-fidelity Prototype - Iteration 2

The high-fidelity prototype consisted of eleven different views and they can all be found in Appendix F. Some of the views are presented here, for example Figure 5.2.3 which is the starting page. By using the menu the user could either create a new session, replay a session, create a new setup, change a setup or exit the program. Depending on which page the user goes to the text in the top border of the program will be updated. In this figure it only says “TrackerServer”. To get help the user could click on the help-menu and receive a user manual. The starting page is only seen once when the program is started and when the user chooses to move to another part of the program the user cannot go back to this view.

![Figure 5.4: Starting page for the high-fidelity prototype.](image)

When a new setup-file is created the user will first be presented with a simple view asking for the number of trackers and slaves to be created. The next view will ask for information about the trackers and slaves, such as their position, attitudes and port and IP-addresses and it looks very similar to the view in Figure 5.2.3. The only difference is an add button instead of a save button, and the option to connect the trackers and slaves is not visible. The IDs cannot be chosen by the user since they are automatically defined and increased for every new tracker or slave that is created. When the user has entered all needed data and pushed the add button the view in Figure 5.2.3 will be presented. In this view the user can connect the trackers and slaves and save the setup-file by naming it. At this stage the data fields are disabled and the user can change the entered data by pushing the change button, but the number of trackers and slaves cannot be changed. To perform such a change, the user needs to create a whole new setup-file.

When changing a setup, the first view only contains a simple button which allows the user to choose a specific setup-file from the File Manager. The next view is basically the same as the one in Figure 5.2.3 except for the change buttons which are not visible and all the data entries are enabled. Also, in the top border of this view the whole file directory to the chosen setup-file is displayed.

If the user chooses to create a new session, the view in Figure 5.2.3 will be presented. The user is asked to name the session, choose a directory where the files produced during the session will be saved and also choose which setup-file to use. When the user has entered all this information the second view will appear which only displays the data in the setup-
5.2. Implementation and Evaluation

Figure 5.5: The last page presented when creating a new setup.

Figure 5.6: The first page when creating a new session.

file. From here, the user can push the continue button and will see the third view which is presented in figure 5.2.3.

For each tracker and each slave a halfcircle will be created with two pointers, one for the azimuth values and one for the elevation values. When the session is started these pointers will be updated in real time and display the changes to the azimuth and elevation values. But before the session can be started the user has to create and name buttons for event registration. The user can choose to not create any buttons, or as much as ten buttons. These buttons can then be pushed when the session is started to indicate that an event happened, for example, if the tracker lost track of the target. Each push will be registered in a separate file with the
5.2. Implementation and Evaluation

Figure 5.7: The third and final page when creating a new session.

exact time and the name of the button. In the bottom right corner a red circle for every tracker will be created. Each circle has the tracker’s ID in it. These buttons will change to green when the trackers have found the target during a session. Basically, the red colour tells the user that the tracker has not found the target, while the green colour tells the user that the tracker has found the target and is now following the target. In the bottom left corner are the buttons for starting, stopping and pausing the session.

When replaying a session, the user is first presented with a simple view containing a button which allows the user to choose a specific track log (from a previous session) from the file manager. The track log contains data which has been saved during a previous session. The next view is very similar to the view in Figure 5.2.3 apart from the event-registration in the top right corner which is replaced by a digital clock. This digital clock starts to run when the user pushes the start button and the replayed session is running. The clock is showing the actual time when the data in the track log was first registered.

5.2.4 Evaluation: User Testing - Iteration 2

The results from the user testing is a table of number of errors the users made for each task, a table over the success rate for each task and also a summary of the answers to the questions asked and the noticed behaviour of the users.

Table 5.3: Total number of times an error occurred during user testing in iteration 2.

<table>
<thead>
<tr>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

In Table 5.3 the total number of times an error occurred are presented. In this context the following types of actions were counted as errors: pushing the wrong type of button, trying to interact with parts of the interface which was not made for interaction, moving to a page which was not relevant for solving the specific task and performing an action which led to a popup window displaying either a warning or a reminder/information.
5.2. Implementation and Evaluation

Table 5.4: Success rate for each task during user testing in iteration 2.

<table>
<thead>
<tr>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Successfully</td>
<td>X</td>
<td>XXXX</td>
<td>XXXX</td>
</tr>
<tr>
<td>completed task</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Successfully</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>completed task</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with minor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>guidance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partially</td>
<td></td>
<td>XXX</td>
<td></td>
</tr>
<tr>
<td>completed task</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not completed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>task</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>due to time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constraint</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not completed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>task</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Success rate</td>
<td>25%</td>
<td>100%</td>
<td>75%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>100%</td>
</tr>
</tbody>
</table>

In Table 5.4 the success rate for each task can be seen. Each X represents the outcome of one user performing one task. A task was seen as successfully completed if the user was straightforward and only performed the necessary actions for the task and the test moderator did not have to provide any guidance. If the test moderator noticed uncertainty from the user and with minor interaction got the user back on the right track, the task was marked as successfully completed with minor guidance, if the guidance resulted in the user reaching the end of the task. To mark a task as partially completed the user did not follow the steps in the task description correctly, might have explored other parts of the interface than the relevant ones, but still ended up with a correct result. At the bottom of the table the success rate is presented, which was calculated by only dividing the number of successfully completed task by the total number of users and multiplying by 100.

User 1

The user thought it was easy to work with the interface and to understand how to perform most of the instructions from the tasks. Regarding how easy it was to make a mistake in any of the four tasks the user did not experience it to be more easy nor more difficult than in any other program used. The user noticed doing one mistake during the test session but said it was easy to correct it.

One part of the interface did confuse the user and it was regarding the part in the first task that said to connect the tracker and the slave. The user would have preferred a heading saying “connection” to make it more obvious that the checkbox’s intention was to connect the tracker and the slave. However, the user did complete the task and connected the tracker and the slave successfully even though some confusion was expressed during the execution of the task. Further comments regarded improvements of the view for running a created session. First of all the user wanted to have a digital clock somewhere in the view when starting the new session and having the clock showing the real time. Secondly, the user wanted some type of response when pushing any of the buttons for event-registration. For example a list with the time and the name of the button that has been pushed. Thirdly, the halfcircles and the two pointers showing the azimuth and elevation values could be removed and instead display the numerical values only.

User 2

The user thought it was easy to understand the interface and that it was structured in a logical way. Furthermore, the user said the tasks were simple and did not to experience any difficulties in performing them or interacting with the interface.

No specific part in the interface did seem to confuse the user, but a minor mistake happened in the first task. The user was supposed to perform a change to the setup before saving it, but instead the user saved the setup and then used the menu and moved to the view for changing setup to perform the changes. Afterwards the user said such a mistake would not
happen if you read the whole task description carefully before you start. Regarding improvements the user had opinions about the view for replaying a session. The first suggestion was to be able to move backwards or forwards to a specific time slot and also to replay at a higher speed. The second suggestion was to somehow display the events and when they would occur during the replay.

User 3

Overall, the user thought the interface was really good, easy to work with and easy to understand. Regarding the risk of mistakes, the user did not think it was easy to perform faulty actions and make mistakes during any of the four tasks. However, the part where the user was told to start the session (in task number three) did confuse the user at first. The confusion resulted in the user moving away from the current view by using the menu and exploring other parts of the prototype. Eventually the user found the way back to complete the task and later said it all happened because of eager to perform the task. The user also added that if it would have been a real flight test environment it would probably not have happened because there had been enough time to complete the task in peace.

Several suggestions for improvement were discussed and the first one regarded the field for connecting trackers and slaves. Instead of only having the ID of the object and the text “tracker” or “slave”, one could also add the type of each object to minimise the risk of connecting wrong trackers and slaves. A type could be “Tracke” for a tracker and “Golden Eye” for a slave. Another suggestion was some form of response when pushing the buttons for event-registration when running a session.

Regarding the view where the user could create a new setup-file, one suggestion about the entry fields for latitude and longitude values was expressed. As for now, the prototype accepted the values in decimal degrees. But apart from decimal degrees, latitude and longitude can be represented in degrees and decimal numbers, and degrees, minutes and seconds. Therefore the user wanted a live conversion of the values, in all different representations, displayed somewhere in the view. Meaning, when the user is entering a latitude value with the decimal degrees representation, the same value but in degrees and decimal numbers, and in degrees, minutes and seconds will also be displayed somewhere in the view, as long as the marker still is in the entry field for the latitude value. The same goes for the longitude value.

The next suggestion was to visualise the data that is sent and received during a session by plotting the values in a coordinate system. The azimuth and elevation values for both the tracker and the slave can be converted to x- and y-values in separate coordinate systems. By creating a line between every successive x- and y-point a pattern will be formed and these two patterns can be compared to indicate if the tracker and the slave seems to be pointing at and following the same target.

Lastly, the user thought it would be nice with a bigger fontsize for all numbers and letters.

User 4

The user thought the interface was pretty good and straight forward. It was not especially easy to make mistakes. Even though some confusion appeared the user said it could have been because the own expectations did not fully align with the interface or how the instructions for the tasks were formulated.

The first suggestion for improvement regarded the view for creating a new session and when the user was told to choose a directory. The user did not think it should be mandatory to choose a directory, but instead have a default directory which could be used. As for now, when a directory is chosen the name is displayed, but the user thought the default directory should be displayed at first, before the user has chosen something else. In general, the user wanted the default or the chosen directory to be displayed in the top border of the interface at all times.
Furthermore, the user wanted to be able to choose the input format for latitude and longitude and if a default format is used it should be displayed somewhere. The user also would like to be able to use keyboard shortcuts, such as Ctrl-S for saving instead of pushing a save button in the interface. Lastly, a more distinct separation of the trackers and slaves in the view for creating and changing a setup-file was desired. For example, assigning the trackers with ID T1, T2, T3 and so on and the slaves with ID S1, S2, S3 and so on. A second example was to either have a clear line between the trackers and the slaves or to have different colours for the entry fields. One colour for the trackers input fields and another colour for the slaves input fields.
6 Discussion

This chapter is divided into three main sections. Firstly, the results from the pre-study, the first iteration and the second iteration will be discussed. Secondly, the method will be discussed with focus on the weaker parts and how these parts could have affected the result. The method discussion is divided into smaller sections touching on specific parts in the method. Furthermore, the validity and reliability of the method will also be brought up, as well as some source criticism. Lastly, the work will be discussed with a wider perspective, both ethical and societal.

6.1 Results

In this section the results from both the pre-study and the two iterations with implementation and evaluation will be discussed. The results will be related to material covered in the theory chapter to some extent.

6.1.1 Pre-study

During the requirements elicitation process both interviews and prototyping were used. The amount of useful information retrieved from the two different techniques was significantly higher for the prototyping part of each session. The participants’ answers to the interview questions were still useful to get a wider perspective and understanding, but the prototyping resulted in a lot more comments and useful information for the development of the interface of the program. This aligns well with Vijayan and Rajus [16] results saying users have difficulties in explaining their requirements using only words. Users tend to find it easier to express their wants and needs when being presented with a prototype of the actual system.

6.1.2 Implementation and Evaluation

The implementation in the first iteration was inspired by the design principle ABS which in the study [2] showed to decrease the amount of errors and improve the task-completion. However, the results from the evaluation in the first iteration shows a total number of errors which is relatively low and a success rate which is also relatively low. Therefore it is not
obvious to draw any conclusion about the effect of using the ABS design principle for designing user interfaces, which also can be strengthened by the researchers themselves saying the technique needs to be tested more for it to be widely accepted. But one can at least argue about the fact that the results from the first iteration in this thesis showed a low amount of errors. Regarding the success rate, it is important to have in mind the fact that only the tasks which were marked as “successfully completed” were used in the calculation, which means the ones called “successfully completed with minor guidance” were not considered and could thus have affected the outcome. A second aspect is the ability to create and formulate good tasks for the users, which is further discussed in section 6.2.2 and the effect it could have on the users ability to successfully complete a task. Thirdly, all the users said they did not experience any difficulties in understanding or performing any of the tasks. These comments are also important to have in mind when discussing the impact ABS could have in the process since the success rate could be misleading.

An error could be categorised as a slip, which is when the user has the correct goal but performs wrong actions, or as a mistake, which is when the user both have wrong goal and performs wrong actions. According to Nielsen, errors could be separated according to their impact. Either an error occurs which the user notices and can correct, or the error goes unnoticed and can destroy the user’s work later. In this thesis the errors were not categorised in any way which might have led to some difficulties in analysing the results. By both having the errors categorised based on the severity level and having the total number of errors could have given a more depth to the discussion. Furthermore, it is possible that the total number of errors is not completely representative from both iterations. Especially from the first iteration which included the paper prototype, since it was difficult to both notice when the user made slips or mistakes and provide the correct view of the interface each time the user interacted with some part. Further discussion regarding the role of the test moderator during the evaluation can be read in section 6.2.5.

In this thesis, each test session was held in similar meeting rooms for a duration of maximum one hour. According to the researcher Norman, the most common reason for why errors occur is because users are put in environments where they are expected to act in unnatural ways. This could partially align with a comment made by user 3 in the second iteration when a mistake occurred. The user said that if it would have been a real flight test environment this mistake would probably not have happened because they would have had enough time to complete the task in peace. Based on this comment and Normans statement one can argue that some of the mistakes made by the users during these test sessions might have occurred because the environment and the tasks were in some way unnatural. Maybe the users felt more stress and pressure to complete the tasks than they would feel if it would have been a real situation.

According to research made by Walker, Takayama and Landay, users tend to leave more comments on a computer-based prototype compared to a paper prototype (although there were no difference in the number of discovered usability problems). This result does not seem to align with the outcomes from this thesis since both prototypes seemed to result in mainly the same amount of comments. The fact that the paper prototype was used before the computer-based prototype could have affected the result. Another aspect could be that the computer-based prototype in the second iteration did align more with the requirements of the users and thus did not result in a greater amount of comments.

Both Nielsen and Quesenbery describe how a system should handle errors and what it means for it to be error tolerant. They both determine that if a user makes an error the system should be designed such as the user can easily recover from them. This aligns really well with a comment from user 1 from the second iteration, saying that it was easy to
correct a mistake that happened during the interaction with the interface. Moreover, none of the tasks performed in the first or in the second iteration was marked as not completed (see table 5.2 and table 5.4). They either were successfully completed (some with minor guidance) or partially completed which still meant they had reach a correct result. Together with the comment from user 1, this result could indicate that the interface could help the users to recover from their own mistakes.

In the end the users seemed to appreciate the interface and thought it was easy to interact with based on the comments retrieved from the user testing. This final interpretation can be strengthened with the study made by Abad et al [17] saying that using paper prototyping in the requirements elicitation is especially helpful for creating a more user-friendly and intuitive product.

6.2 Method

In this section a discussion regarding the chosen method will be held. Focus will be on weaker parts of the method and how it could have affected the results. Relevant theory will be mentioned and used to strengthen the analysis. Furthermore, the validity and reliability of the study will be discussed as well as the sources used for finding relevant theory.

6.2.1 Interview Questions for Requirements Elicitation

Using interviews as one approach for requirements elicitation is said to be very useful since it results in a lot of data, but the quality of the data depends on the interaction between the participants [11]. In this thesis, the ability to formulate good interview questions could also be questioned based on the study performed by Hadar, Soffer and Kenzi [13]. Their results show that participants with domain knowledge found it easier to formulate interview questions, while the participants without domain knowledge experienced some difficulties in understanding the customer and the terminology used. This thesis started with a pre-study where domain knowledge was retrieved to facilitate the formulation of interview questions but also to get a better understanding of how and when the interface for the program would be used. However, the time-period spent on researching for relevant domain knowledge might have been too short and the level of domain knowledge you could retrieve as a researcher could depend on the specific domain. For example, it is a difference between a person only gathering information and theory, compared to a person who actually works in the specific domain and thus learns the theory and terminology in a different way. Another aspect regarding the interview questions is that they were never tested in some sort of pilot study before the requirements elicitation part. Such a pilot study could have been used to get feedback on the questions, how they were formulated and if some question was missing or difficult to understand.

This could have affected the results in such a way that not all the important information was revealed at an early stage, which might have lead to some requirements not being noticed. A potential lack of enough domain knowledge could also have led to more misunderstandings which could affect the design of the prototype and thereafter the formulating of correct and relevant tasks for the user testing.

6.2.2 Number of Users for User Testing

During the first iteration of user testing three users were used and during the second iteration four users were used. The number of users were mainly based on the availability. But one can still discuss if the number of users were enough for this thesis. The thinking aloud technique, which was used together with user testing, is said to require a minimum of three test users [18] while the number of users for user testing is a bit more diverse. The known "industry
standard” is saying five participants is the number to aim for [22], while other research conclude that five participants is nowhere near enough to discover enough usability problems [20] [23]. Adding to this, some researches indicate that the most important is not the number of participants but rather the number and formualtion of tasks for them to perform during the tests [24] [25]. However, the fact that this thesis had three respectively four participants in the user testing process could have affected the result in such a way that some usability problems never were found. Especially since this thesis was said to focus more on the task creation rather than on the amount of participants. The number of tasks and how they were created and formulated could also be another aspect of not finding enough usability problems. If more tasks had been created for each iteration of user testing, or if they had been structured and formulated in a different way, it might have affected the results such as the number of errors performed by the users or their feedback regarding the overall impression of the interface.

6.2.3 Number of Iterations

To combine evaluation and iterative design is a good idea because a solution that solves one usability problem can introduce another. Therefore, with an iterative approach the design can be refined several times and hopefully create a better end product. [3] This thesis was divided into a pre-study and two iterations which included implementation and evaluation. The choice of only having two iterations was because of time constraints, but one can discuss if the number of iterations was too few and if adding one would have made a difference. Since Nielsen [3] talks about the positive effects of iteratively create the design, one could argue that more iterations would create a better interface design. Maybe one more iteration in this thesis would have revealed more usability problems and thus resulted in a better end product.

The number of iterations could also include the requirements elicitation process. Vijayan and Raju [16] used paper prototyping in the elicitation process in their study, and the results showed that the participants needed 1-3 iterations to complete the process. In this thesis paper prototyping was used, together with interviews, during the requirements elicitation but only for one iteration. Maybe it would have been better to at least have two iterations to gather more requirements. It could have affected the results such as creating more requirements or refining the ones found in the first iteration.

6.2.4 User Testing as Evaluation Method

Two studies [20] [21] have compared user testing and heuristic evaluation as techniques for identifying usability problems. They both reach the conclusion saying the two techniques are rather complimentary than competing and could be used at different stages in the development process. Although these studies were conducted on websites, the idea could still be suitable for this thesis. For example, a heuristic evaluation could have been used in the beginning of this thesis, followed by user testing. Holzinger [18] categorises herustic evaluation as an inspection method and the idea is to evaluate the system against known usability standard. This approach is quite difficult from how user testing is performed, and maybe that is one reason why both the studies [20] [21] found user testing and heuristic evaluation to detect different types and varying amounts of usability problems.

By only using user testing (although together with the think aloud approach) this could have resulted in detecting fewer usability problems. With two different techniques, which is said to be complementary, one could have investigated new parts of the interface and found other ways to execute certain tasks. The combination could have covered a greater spectrum and thus produced an even more usable product.
6.2. Method

6.2.5 Role of the Test Moderator

During a test session in this thesis, apart from the user the only other participant was the test moderator. This resulted in the test moderator being responsible for all sorts of tasks which otherwise could have been spread among more than one person. For example both giving the user general instructions in the beginning of each session and before each task, as well as noticing the user’s behaviour when performing the tasks and when and how the user made slips and mistakes. This could possibly have affected the result such as not providing the exact same information to all users during all tests or missing to notice when a user made a slip or mistake. Although the think aloud technique was used and all sessions were audio recorded (the last iteration the computer screen was even video recorded when the users interacted with the prototype), it would probably have been easier to handle all the tasks if more than one person had been responsible.

Furthermore, one can discuss the possible impact the test moderator had on the test users. Both in the aspect of how many mistakes the users made when performing the tasks and also in how they answered the questions asked after each test session. Maybe some of the users felt more stressed by having the test moderator observing them and maybe some of the answers were given because the users thought that was what the test moderator would like to hear.

6.2.6 Separate User Tests

The structure of the user testing in this thesis was to have separate sessions for each user. The main reason was because it would be easier for the test moderator to observe one user at a time, and also because the users would not be able to influence each other and maybe feel more free to delve into new thoughts and ideas. However, the experience for the test moderator was sometimes a bit confusing and challenging when the users seemed to have contradicting views on certain things, such as the actual aim of the program or what features were needed. This could have affected the results as in how to determine what to prioritise and how to actually design the interface. Therefore it might have been appropriate to also have some sort of brainstorming session in the end of each iteration where all the users could gather and discuss their opinions about the prototypes. This brainstorming could have solved some misunderstandings, both between the users themselves and between user and test moderator.

6.2.7 Validity and Reliability

To enhance the validity and reliability of a study one can use triangulation methods by using several kinds of data and combining different methods [31]. In this thesis interviews and prototyping has been used during the requirements elicitation, and user testing and the thinking aloud technique has been used during the two iterations for evaluation. Moreover, both qualitative and quantitative data has been collected such as the users comments and answers to predefined questions, number of errors made during the interaction with the interface and the success rate for each user and each task. By using these combinations of techniques and approaches it strengthens the validity and reliability of this thesis.

Validity is about making sure you really are measuring what you are supposed to be measuring and since the developed program is intended to be used in a flight test environment only users with the appropriate domain knowledge were used during the requirements elicitation as well as the user tests. The aim for developing the prototype during the two iterations was for it to be as alike the final product as possible, especially the high-fidelity prototype. This also strengthens the validity of the study. Moreover, since the research question included evaluating the usability and error rate of the interface for the program, the questions asked at the end of each test session can strengthen the fact that the right thing was being measured. The questions can be found in chapter 4.
Reliability regards the possibility to reach the same results once again if the same study was conducted. The method chapter in this thesis was written with as much details as possible to enhance the reliability. In the appendix both the structure and the interview questions for the requirements elicitation can be found, as well as the test plan used for all the user tests in iteration 1 and iteration 2. Each user test was conducted in the same environment each time with the purpose of affecting the results as little as possible. However there are several aspects which could have affected the reliability that needs to be discussed.

First of, the categories used for counting the number of errors each user made during each task. Another researcher might not mark a specific action by the user as an error and thus interpret the categories in a different way. This might lead to weaken the reliability of this thesis. A second aspect is about the formulation of some of the tasks for the user testing in iteration 1. The tasks were basically the same for each user, apart from the switching between sequential or parallel view for task 1 and task 4. Since the users did not get the exact same tasks, the results might be affected by which person that got which task. Some test persons might have understood the formulation of these tasks better than others.

6.2.8 Source Criticism

In this thesis the aim was to use well-cited scientific papers as much as possible. Well-cited papers are believed to have a higher credibility and many scientific papers are peer-reviewed before being published. Google Scholar was mostly used for the search of these kinds of papers and to some extent the databases IEEE, ACM and Springer. By using Google Scholar one can easily see how many citations a specific article has, in which articles it has been cited and also find other related articles. But even though an article has many citations it does not necessarily mean it is better than an article with lower amount of citations. A citation is usually a positive indication, but it is also important to keep in mind that it could appear in a negative context. Since not only Google Scholar was used, but also IEEE, ACM and Springer, the idea was to increase the possibility of finding good and relevant scientific papers.

Since the area of technology and human-computer interaction is evolving the choice of using mostly scientific papers felt relevant. However, if no relevant scientific paper could be found a book has been chosen as a second option.

During the literature study the aim was also to use scientific papers that had been written and published recently. Mainly because of the same motivation as mentioned earlier, the area of technology and human-computer interaction is evolving. But sometimes no such relevant papers could be found and to find a balance older scientific papers were accepted as long as they were well-cited. If a paper was published recently it could also be accepted even though it was not well-cited, as long as its content was relevant.

6.3 The work in a wider context

In this thesis project the ethical and societal aspects have been considered when involving users in the requirements elicitation and the evaluation processes during the two iterations. Every meeting with the users, whether it was an interview for gathering requirements or a test session for user testing, started with information regarding the principles for ethical research involving humans. The users were told that their participation was completely voluntary, they could cancel and leave the session whenever they wanted, the results from each session was only going to be used in this thesis and they were going to be anonymous. Each session was also recorded, but only after permission had been given by the user. Even though some of the users participated in all three parts of the process (the interview in the requirements elicitation, the user test in iteration 1 and the user test in iteration 2) the same information was still provided before each session.
7 Conclusion

This chapter is divided into two sections. The first one summarises the purpose of this thesis and the answer to the research question. The second part discusses future work and presents relevant ideas.

7.1 Research Question

The aim of this thesis was to examine how you can both design and evaluate a user interface with regards to usability and low error rate. The research question was formulated as follows:

*How can you design and evaluate a user interface for a program, used in flight tests of electronic warfare systems, with regards to usability in terms of low error rate?*

The design phase was conducted in an iterative manner, just as Nielsen [3] mentions as one of the stages in the usability engineering lifecycle. By iterative design the developer can receive continuous feedback and refine the product in several stages, which can be seen in how the low-fidelity prototype changes into the high-fidelity prototype in this thesis. Nielsen also provides some guidelines in how to create good error messages when creating a user interface, and these guidelines were used during the design phase as well as the guidelines provided by Quesenbery [32] on how to prevent errors. Lastly, the anchor-based subgoaling (ABS) [2] did inspire the creation of the prototypes. However, no general conclusion can be drawn regarding the use of ABS since the researcher themselves determine that more studies has to be performed with the technique for it to be widely accepted.

The evaluation of the user interface was performed with user testing and with users who were representative of the target audience. The thinking aloud technique was used and the users were encouraged to explain their thoughts when performing the tests and interacting with the prototypes. This helped in understanding when and why the users were confused and did not know how to interact with the prototypes. A few complementary questions regarding the users general experience with the prototypes were also asked at the end of each session to get a deeper understanding since the thinking aloud technique might not reveal all the users’ thoughts. Furthermore, notes were taken to count the number of errors each user made and if they completed the task or not. This data indicated how well the
interface prevented the users from performing errors and how well the interface helped the users to still complete the tasks.

The results from this thesis indicate that the users were satisfied with the interface, did few errors when interacting with the interface and when errors occurred they could recover from them. This aligns with both Nielsen’s [3] and Quesenbery’s [32] definition of an error tolerant system. Therefore a conclusion from this thesis can be drawn that all the steps taken in the design and evaluation phase has affected the results and created a user interface with regards to usability in terms of low error rate.

7.2 Future Work

In chapter 6 the role of the test moderator is discussed and however it might have affected the users and their performance. Based on that, an idea for future work could be to structure the user tests in such a way that no test moderator is present but instead the whole session is video recorded. This type of user test could than be compared to the ordinary ones which has a test moderator observing the whole session, and the results could be analysed to see if users tend to perform more errors when being physically observed or not.

Another idea for future work could be to investigate more in the ABS design principle. This thesis was only inspired by the design principle, but it could be interesting to perform a study which focuses much more on it. Especially since the researches themselves has said their is not enough studies performed in the area. It would be interesting to design an interface for a whole program were the user can perform several different tasks, and use the ABS for all these tasks.


Appendices
Appendix: Requirements Elicitation - Interview Guide

This is a structure for the interview and prototyping session held during the requirements elicitation process:

- Ask about permission to record the session.
- Inform the participant about the ethical principles for conducting research with human participants, including explaining the aim of the study.
- Explain the approach for this session. First interview questions and then a prototype.
- Encourage the participant to ask questions if anything is unclear.
- Allow for pauses between the questions, do not stress through either the questions or the answers and do not interrupt the participant.
- Perform the interview.
- End the interview by asking if the participant has anything else to add.
- Make it clear that the interview has ended and move on to the prototype.
- Explain that this prototype is mainly used for gathering more requirements. It does not necessarily indicate how the final user interface will be designed. Also be clear that this prototype is mainly based on the initial requirements.
- Let the participant examine the prototype and ask for thoughts, ideas and opinions.
- Make a clear ending of the session, thank for the participation and encourage the participant to get in touch if any questions or thoughts emerge afterwards.
Appendix: Requirements
Elicitation - Interview Questions

General questions not related to the program

- What is your profession?
- How long have you worked here?
- Have you worked in the same area of expertise during your whole time here?

General and specific questions related to the program

- Have you used the current program?
  - If yes, continue.
  - If no, formulate the following questions more hypothetically. Such as “How do you think...” or “When do you think...”.

- When do you use the program?
  - In which environment do you use the program?

- How do you use the program?
  - Which functions do you use in the program?
  - How often do you use the program?

- What do you think is important to be able to use the program for?
  - How would you prioritise them? (If several tasks/examples/purposes are presented)

- What challenges can you see with the use of this program...
  - ... based on what types of tasks to be solved?
  - ... based on different environments?
  - ... based on the interaction with the user interface?
• What do you like about the current user interface of the program? (Only asked if the participant has worked with the current program)

• What do you like less about the current user interface of the program? (Only asked if the participant has worked with the current program)
Appendix: Requirements Elicitation - Paper Prototype

Figure C.1: Paper prototype for the requirements elicitation - first view.
Figure C.2: Paper prototype for the requirements elicitation - second view.

Figure C.3: Paper prototype for the requirements elicitation - third view.
Appendix: Test Plan for User Testing

This test plan was developed based on the guidelines from the *Handbook of usability testing: how to plan, design and conduct effective tests* [19].

- **Purpose, goals and objectives of the test** - the purpose of this usability testing is to see how well users with relevant domain knowledge perform when using the interface and tries to complete representative tasks. The main aspect is to see how easy it is to make slips and mistakes which leads to errors, as well as how many errors users perform.

- **Research questions**
  - How difficult is it to complete task X?
  - How easy is it to make a mistake in task X?
  - How well did the user complete task X?
  - How many slips and mistakes did the user make when performing task X?

- **Participant characteristics**
  - Domain knowledge in radar and EW.
  - Have used or seen the old program before.

- **Method (test design)** - this list presents how the test session will be structured:
  - Ask for permission to audio record the test session.
  - Inform the participant of the purpose of the study and the ethical principles for conducting research with human participants.
  - Explain the structure of the session, the role of the test moderator, the importance of thinking out loud, the time constraint on the tasks and the blank-page technique.
  - Start the session. Present one task to the participant at a time.
  - Take notes (as the recording is only audio) when the user makes slips or mistakes or seems to experience difficulties in understanding the tasks or the interface.
  - When all tasks have been performed, ask a few predefined questions related to the tasks and the interface.
• Make it clear that the testing is over and ask if the participant has any questions.

• Thank for the participation and end the session.

**Task list for iteration 1**


2. Change the position of the slave to lat: 58.383353 och long: 15.431891.

3. Start the session. Mark that event three happened. Stop the session.


5. Replay a test session. Close the replay and exit the program.

The first task is supposed to be started when the state of the interface is the start-up view. The second task starts with the view where all the entered information is displayed and where the user can choose to change or save the setup. The third task starts with the view that only has the start, stop and pause buttons and the shortcut buttons. Lastly, the fourth and fifth tasks both start with the blank page view which only has the two buttons start and manual.

Task number 1 and number 4 will vary in their description. Either they will tell the participant to use the sequential or the parallel view when creating a new session. The idea is that if the first task is said to be sequential then the fourth task is said to be parallel, and vice versa. This to be sure that each participant will be presented and familiar with both types of views. The order of task number 1 and task number 4 will also vary, since both regards creating a new session but in different ways. All the other tasks will have the same order during all user testing sessions.

A task is seen as successfully completed when all the subtasks have been performed, in the right order. A task can be seen as partially completed if some of the subtasks were not performed or if the subtasks were completed in different orders. If a user spends more than five minutes on a task it will be marked as uncompleted and the user will be presented with the next task instead.

**Task list for iteration 2**

1. Create a new setup with one tracker and one slave with the following information: Typ: Tracke, Lat: 58.388, Long: 15.511, Höjd: 80, Yaw: 0, Pitch: 0, Roll: 0, Port: 4020, IP-adress: 127.0.0.255 and a slave with the information: Typ: Golden Eye, Lat: 58.487785, Long: 16.319133, Höjd: 80, Yaw: 0, Pitch: 0, Roll: 0, Port: 5020, IP-adress: 127.0.0.1. Connect the tracker and the slave. Change the height of the slave to 50 and save the setup as “test001”.

2. Perform changes to the setup-file “test002”. Change the position for the tracker with ID 2 to: Lat: 59.611277 and Long: 18.080662. Save the setup-file as “changesTest002”.

3. Create a new session and name it to “session003”. Choose the current map directory. Load the setup-file “test004” and continue. Create and name three buttons for event-registration. Start the session. Mark that any of the three events happened and then stop the session.

4. Replay the session named “yesterdaysTest007” by choosing the associated track log. Start the replay. Wait for the replayed session to end (around 30 seconds) and then exit the whole program.
The first task is supposed to be started when the state of the interface is the start-up view. The rest of the tasks are performed one after another, without changing the view of the program in between.

A task is seen as successfully completed when all the subtasks have been performed, in the right order. A task can be seen as partially completed if some of the subtasks were not performed or if the subtasks were completed in different orders. If a user spends more than five minutes on a task it will be marked as uncompleted and the user will be presented with the next task instead.

- **Test environment, equipment and logistics** - each user test will be performed in the same environment, a smaller conference room. For the first test the users will interact with a paper prototype and for the second test the users will interact with a Windows PC and an implemented interface. All test sessions will be audio recorded. If possible, the second test will use a recording program for the computer screen to record the users interaction with the interface. No other equipment will be used.

- **Test moderator role** - the test moderator and the participant are the only two persons participating in a test session. When the participant is solving the tasks, the test moderator is supposed to observe in silence and take detailed notes on the participants behaviour and especially if or when slips and mistakes are made. The test moderator is not supposed to answer any questions regarding the tasks or the design of the interface when the participant is performing the tasks. Minor exceptions are allowed if needed. If one task takes too long to perform (based on a predefined time limit), the test moderator will inform the participant and provide the next task instead.

- **Data to be collected and evaluation measures**
  - Number of tasks successfully completed, with and without help from the test moderator.
  - When the participant made slips and mistakes.
  - Number of errors (error rate). (count of incorrect selection and choices)
  - Answers to the questions asked after all tasks have been performed.
  - Thinking aloud. Quotes from the participants indicating like or dislike.

- **Report contents and presentation** - in the final report, all data gathered during the tests will be presented as the result. Both the qualitative data such as summaries of the audio recordings and the quantitative data such as number of tasks successfully completed and number of errors made.
Appendix: Low-fidelity Prototype - Iteration 1

Figure E.1: Low-fidelity prototype for iteration 1 - start page.
Figure E.2: Low-fidelity prototype for iteration 1 - second page.

Figure E.3: Low-fidelity prototype for iteration 1 - third page.
Figure E.4: Low-fidelity prototype for iteration 1 - fourth page.

Figure E.5: Low-fidelity prototype for iteration 1 - fifth page.
Figure E.6: Low-fidelity prototype for iteration 1 - sixth page.

Figure E.7: Low-fidelity prototype for iteration 1 - seventh page.
Figure E.8: Low-fidelity prototype for iteration 1 - eighth page.

Figure E.9: Low-fidelity prototype for iteration 1 - ninth page.
Figure E.10: Low-fidelity prototype for iteration 1 - tenth page.

Figure E.11: Low-fidelity prototype for iteration 1 - eleventh page.
Figure E.12: Low-fidelity prototype for iteration 1 - twelfth page.

Figure E.13: Low-fidelity prototype for iteration 1 - thirteenth page.
Figure E.14: Low-fidelity prototype for iteration 1 - popup windows and minor objects.
Figure F.1: High-fidelity prototype for iteration 2 - start page.
Figure F.2: High-fidelity prototype for iteration 2 - new session: first page.

Figure F.3: High-fidelity prototype for iteration 2 - new session: second page.
Figure F.4: High-fidelity prototype for iteration 2 - new session: third page.

Figure F.5: High-fidelity prototype for iteration 2 - replay session: first page.
Figure F.6: High-fidelity prototype for iteration 2 - replay session: second page.

Figure F.7: High-fidelity prototype for iteration 2 - new setup: first page.
Figure F.8: High-fidelity prototype for iteration 2 - new setup: second page.

Figure F.9: High-fidelity prototype for iteration 2 - new setup: third page.
Figure F.10: High-fidelity prototype for iteration 2 - change setup: first page.

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Latitude</th>
<th>Longitude</th>
<th>Height</th>
<th>Yaw</th>
<th>Pitch</th>
<th>Roll</th>
<th>Port</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Truck</td>
<td>35.05</td>
<td>114.52</td>
<td>10.0</td>
<td>90.0</td>
<td>80.0</td>
<td>100</td>
<td>400</td>
<td>192.168.1.23</td>
</tr>
</tbody>
</table>

Figure F.11: High-fidelity prototype for iteration 2 - change setup: second page.

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Latitude</th>
<th>Longitude</th>
<th>Height</th>
<th>Yaw</th>
<th>Pitch</th>
<th>Roll</th>
<th>Port</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GoldenEye</td>
<td>12.60</td>
<td>114.30</td>
<td>10.0</td>
<td>90.0</td>
<td>80.0</td>
<td>100</td>
<td>400</td>
<td>192.168.1.23</td>
</tr>
</tbody>
</table>

Tracker1
- Step1
- Step2

Opcua scrm