Comparing two heuristic evaluation methods and validating with usability test methods

- Applying usability evaluation on a simple website

A study at Nordiska Entreprenadsystem AB

Michael Sohl

Supervisor, Ola Leifler
Examiner, Cyrille Berger
Abstract

In the field of usability engineering, researchers are looking for cheap usability evaluation methods that can unveil most of issues. In this study, an IT company asks for an administration tool, realized as a webpage. The combinational usage of usability inspection and usability testing has been proven to give good evaluation feedback and was therefore applied in development of the support tool, in the hopes of achieving high satisfaction. The author tried during the study to perform the evaluation methods as cheaply as possible and still obtain a good result after each iteration. There were signs that a cheap and less formal method trades time for quality. Even though, an increase of satisfaction was observed.

Another significant question that was central to the study was the comparison between the industry leading heuristic evaluation with Nielsen’s heuristics and Gerhardt-Powals principles, created from the cognitive sciences. Only one previous study was found making this comparison which made it interesting to see if the same result would be reached.

The result showed an increase of 6.2% measured from Standard Usability Scale (SUS) -questionnaires between the first and second development iteration. The notion of an increased satisfaction was also indicated from heuristic evaluation which between iteration one and two yielded 40% less faults and the number of severe faults went from 20 to 2. Usability testing also showed on average decline in negative comments about the user interface.

Comparing Nielsen and Gerhardt-Powals yielded a similar number of faults found, in line with the results of (Hvannberg et al., 2007). An observed difference was although that the predicted problems of Gerhardt-Powals had a greater match with real problems found by end-users than Nielsen. On the other hand, Nielsen discovered slightly more severe issues than Gerhardt-Powals.
# Table of Contents

1 Introduction.................................................................................................................... 7
  1.1 Motivation.................................................................................................................... 7
  1.2 Purpose....................................................................................................................... 8
  1.3 Research Questions..................................................................................................... 8
  1.4 Delimitations .............................................................................................................. 8
  1.5 Company directives .................................................................................................... 8
  1.6 Abbreviations ............................................................................................................ 9

2 Background..................................................................................................................... 10
  2.1 Company description .............................................................................................. 10
  2.2 Study context ............................................................................................................ 10

3 Theory................................................................................................................................ 11
  3.1 Usability ...................................................................................................................... 11
  3.2 The User ..................................................................................................................... 12
    3.2.1 Know the user and the tasks ............................................................................. 12
    3.2.2 Prototype ............................................................................................................ 12
    3.2.3 User experience metrics .................................................................................. 13
    3.2.4 When to measure during a product's life cycle? ............................................ 14
    3.2.5 Design processes ............................................................................................... 14
  3.3 Usability Evaluation Methods .................................................................................. 15
    3.3.1 Other methods .................................................................................................... 15
    3.3.2 Previous studies ................................................................................................ 15
    3.3.3 Challenges in evaluation .................................................................................. 17
  3.4 Heuristic evaluation ................................................................................................... 17
    3.4.1 Perform an evaluation ....................................................................................... 18
    3.4.2 Nielsen's 10 ...................................................................................................... 18
    3.4.3 Gerhardt-Powals ............................................................................................ 20
  3.5 Evaluation with end-users ......................................................................................... 21
    3.5.1 Think aloud ........................................................................................................ 22
    3.5.2 Other methods .................................................................................................... 22
  3.6 Web programming frameworks ............................................................................... 22
  3.7 Data collection methods ......................................................................................... 23
    3.7.1 Interview ............................................................................................................ 23
    3.7.2 Observation ...................................................................................................... 24
    3.7.3 Questionnaires ................................................................................................. 24
    3.7.4 Sampling ............................................................................................................ 24
4 Method ........................................................................................................... 25
  4.1 Pre-study .................................................................................................. 25
    4.1.1 Interviews ......................................................................................... 25
    4.1.2 Literature study ................................................................................ 26
    4.1.3 Prototyping ......................................................................................... 26
    4.1.4 Programming ..................................................................................... 27
  4.2 Implementation ....................................................................................... 27
    4.2.1 Design guidelines ............................................................................. 27
  4.3 Heuristic evaluation ............................................................................... 28
    4.3.1 Choice of evaluators ......................................................................... 28
    4.3.2 Nielsen and Gerhardt-Powal ............................................................ 28
    4.3.3 Evaluation session ............................................................................ 30
    4.3.4 Data gathering .................................................................................. 30
    4.3.5 Data filtering .................................................................................... 31
  4.4 Think Aloud ............................................................................................ 31
    4.4.1 Choice of evaluators ......................................................................... 31
    4.4.2 Evaluation session ............................................................................ 32
  4.5 Prepare new iteration ............................................................................ 32
  5 Results ......................................................................................................... 34
    5.1 Pre-study ............................................................................................... 34
    5.2 First iteration ......................................................................................... 36
      5.2.1 System description .......................................................................... 36
      5.2.2 Heuristic evaluation ....................................................................... 38
      5.2.3 End user tests ................................................................................ 40
      5.2.4 SUS-score ...................................................................................... 42
    5.3 Second iteration .................................................................................... 42
      5.3.1 System description .......................................................................... 42
      5.3.2 Heuristic evaluation ....................................................................... 48
      5.3.3 End user tests ................................................................................ 49
      5.3.4 SUS-score ...................................................................................... 51
  6 Discussion ..................................................................................................... 52
    6.1 Result ..................................................................................................... 52
      6.1.1 Pre-study ......................................................................................... 52
      6.1.2 Heuristic evaluation ....................................................................... 52
      6.1.3 SUS-score ...................................................................................... 53
  7 Conclusion .................................................................................................... 55
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Nielsen vs Gerhardt-Powals</td>
<td>55</td>
</tr>
<tr>
<td>7.2</td>
<td>Interface improvement</td>
<td>55</td>
</tr>
</tbody>
</table>
Index of Tables

Table 1: List of abbreviations in the report................................................................. 9
Table 2: Gould & Lewis (1985) list 3 important components that are involved in UCD. .......... 12
Table 3: List of Gerhardt-Powals principles that were used during evaluation. .................... 29
Table 4: List of Nielsen’s heuristics that were used during evaluation. ............................... 29
Table 5: Created tasks for scenarios.................................................................................. 32
Table 6: Requirements that are required to be implemented. The five listed are taken from User Requirements document............................................................... 35
Table 7: Comments of the most severe problems found, and four randomly picked problems of less severity, that were recorded during expert evaluation with Nielsen’s heuristics. ......................... 39
Table 8: Comments of the most severe problems found, and some problems of less severity, that were recorded during expert evaluation with Nielsen’s heuristics. .................................................. 40
Table 9: A sample of some of the comments recorded during TA on the first iteration .......... 40
Table 10: SUS score gathered from after the first iterations TA- session. .............................. 42
Table 11: Comments of the most severe problems found, and some problems of less severity, that were recorded during expert evaluation with Gerhardt-Powals principles. .............................. 49
Table 12: Comments of the most severe problems found, and some problems of less severity, that were recorded during expert evaluation with Nielsen’s heuristics. ............................................. 49
Table 13: A sample of some of the comments recorded during TA on the second iteration ......... 49
Table 14: SUS-score gathered from after the second iteration’s TA- session........................... 51
1 Introduction

In this study two usability evaluation methods were compared. There are several methods in this area, and companies need cheap methods to create highly usable systems. Such systems should facilitate work at companies at which they are introduced. However, today there are many examples of systems that make the daily work of employees difficult instead.

1.1 Motivation

In today’s web- and technique-driven business climate, more and more effort and resources are funnelled into development and design of IT systems. This is the case according to Söderström (2015), that more money than ever before is spent on IT systems. However, researchers in the field of information systems are surprised over how bad and how hard some of the systems are to use. A trend of stress has also been linked to systems with low usability in the workplace.

In this study an administrative support tool was created for the workplace at a company called NEAB. Creating an easy to use user interface, not prone to inducing stress or frustration like many other systems at workplaces, had its design and implementation challenges. Although, measuring whether a user interface has a high or a low usability has an immediate value. Holzinger (2005) claims that the application of Usability Evaluation Methods (UEM) is very likely to lead to higher usability, but many companies are still reluctant to implementing such methods. The reluctance is due to the high costs of hiring of Human Computer Interaction (HCI) experts, allocating laboratory environments and analysing gathered evaluation data. Therefore, there is a motivation for the development of cheap UEMs and for them to be effective and efficient in such a way that developers could use them. In addition, it is also important to raise the awareness among software developers to apply usability methods to improve user interfaces today.

A large portion of the research is focused on usability evaluation and about refining and developing methods. Heuristic Evaluation (HE) is one these, being one of the most used in industries (Holzinger, 2005). This study has reviewed several studies concerning the validation of usability and the comparison of different evaluation methods. Most of the studies have primarily focused on evaluation of user interfaces late in development. Only a few studies have integrated UEMs as a part of development in an iterative implementation process of a user interface from scratch.

NEAB, the company at which the implementation took place expressed the need for a simple administrative-tool to facilitate the work of the employees engaged in customer support. The premise was that the administration tool would improve work efficiency and reduce downtime. To achieve this the user interface needed to reach a high degree of usability. Otherwise an introduced system could have a negative affect instead. In this study UIMs (Usability Inspection Methods) and UTMs (Usability Test Methods) were used to validate the administration tool. The UEMs also acted as a supplement for steering reiterations of the implementation process. The tool was created to help the daily work of NEAB’s customer-support team. The administration tool realized a set of database operations that gave the support team previously restricted access through a webpage interface. The time it takes to conduct usability evaluations was also considered in this study, because of the reluctance showed in the industry today.

Hvannberg, Law & Léruisdóttir (2007) compared Gerhardt-Powal’s principles and Nielsen’s heuristics. The number of studies on evaluating Nielsen’s heuristics are substantial and very few have been conducted by employing Gerhardt-Powal’s principles. In contrast to Nielsen’s heuristics, Gerhardt-Powal’s principles are taken from the cognitive sciences, promoting aspects like situational awareness. Gerhardt-Powals (1996) tested these principles when building a user interface for a submarine’s firing system, on which it was deemed superior to a version built with without the principles. The system
that has been implemented in this study is not as safety critical when, but some aspects of the administration tool were critical to some extent. For example, some actions could result in the loss of customer data. This made Gerhardt-Powals’ principles an interesting candidate for comparing with the industry standard of Nielsen’s heuristics. In addition, with the administration tool the customer support team was for the first time be able to directly alter, add and delete critical database information regarding real customers.

1.2 Purpose

The purpose of this study was to create a webpage for internal usage at an IT-company. The implementation process of the system included both front-end and back-end. After employing an implementation process, steered by applying UEMs in a formative fashion, the goal was to achieve high satisfaction. The tool’s function would ultimately improve the daily work of the support team at the company. Also, methods employed should be cheap and take a small amount of time.

The secondary purpose of this thesis is to compare two UIMs, Nielsen's heuristics and Gerhardt-Powal's principles. Furthermore, matters of interest will include to see how well discovered faults are related to real faults found by end-users performing UTMs. After each iteration, the interface will be subject to evaluation by inspection methods and user testing. The results will inspire redesign to improve the user satisfaction. Feedback received during user tests conducted on the company premise will validate whether if the UIMs can find problems that are relatable to actual problems found by end-users.

1.3 Research Questions

In attempt to achieve the purpose of this thesis a set of research questions has been formulated below. They are presented in order of priority, sorting the research question where the first research question is the main one.

*Can Usability Evaluation Methods be applied in a cheap and formative manner when implementing a website, and still yield a high satisfaction?*

*Is the heuristic evaluation method of ‘Gerhardt-Powals principles’ more effective than that of ‘Nielsen’s 10’ at yielding more usability problems of high severity rating in shorter time?*

1.4 Delimitations

Because the thesis has a limited time frame and that the focus is on usability, there must be a limit for the number of implemented user requirements. Therefore, ranked requirements with regard to priority will be implemented by ability. All employees will not be subject to inquiry since availability and time can be an issue. Customers to the company will not be available for such activities either. Time also sets a limit for how many design iterations are possible. After each iteration, depending on the number of problems found, the most severe will be fixed and then less prioritised.

1.5 Company directives

- The study will be conducted at NEAB in Linköping.
- NEAB is interested in what stakeholders may be interested in an administration tool.
The user interface must be compatible with the existing servers and databases at NEAB. The framework for the user interface’s front-end will be REACT. The framework for the user interface’s back-end will be Django.

1.6 Abbreviations

Below a list of abbreviations is presented.

Table 1: List of abbreviations in the report.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCD</td>
<td>User-centred design</td>
</tr>
<tr>
<td>UX</td>
<td>User experience</td>
</tr>
<tr>
<td>RQ-x</td>
<td>Research question number x</td>
</tr>
<tr>
<td>UEM</td>
<td>Usability Evaluation Method</td>
</tr>
<tr>
<td>UIM</td>
<td>Usability Inspection Method</td>
</tr>
<tr>
<td>UTM</td>
<td>Usability Test Method</td>
</tr>
<tr>
<td>SUS</td>
<td>Standardised Usability Scale</td>
</tr>
<tr>
<td>TA</td>
<td>Think aloud</td>
</tr>
</tbody>
</table>
2 Background

This chapter will shed light on the context of this study, including a short background of the company where the study took place.

2.1 Company description

Nordiska Entreprenadssystem AB (NEAB) is a IT company based in Linköping, Sweden with its office located in central Linköping. NEAB was founded 2011, but the core-business idea that involved a dynamic Enterprise Resource Planning (ERP) -system, dedicated to the construction industry, was first brought into the industry in 1990. Majority shareholder Anders Jacobsson is one of the original founders and informed that the business concept still has not changed, in principle. Even though, platforms and technology have varied since 1990. What makes their product so competitive is dynamicity, modularity, configurability and companies strive to know the customer.

Today, NEAB develops, sells and maintains a single ERP system called NEXT, that 2008 was made entirely cloud-based (Running each customer’s instance of NEXT on NEAB’s servers). Next is deployed as a website for the intent of mobile/tablet usage. A few of its current functionalities include Invoice management, Logging Worked Hours and Map Integration. In the construction industry jobs can vary and diverge from original plans which makes it important for construction companies to document the work done in detail. The reason is basically to provide a transparent representation of the work for which their customers are being charged.

Another reason for the business logic that NEAB enforces, is because the construction industry is difficult to penetrate with IT-solutions, so each installation of NEXT has to meet the needs of each company and tailored accordingly. NEAB currently employs 15-17 people.

The company projects to grow with 50% in the coming years so there is a need for internal revision to cope with future expansion. To add, even though the main aim for NEAB is smaller companies they also have larger clients like NCC and Skanska.

2.2 Study context

As mentioned in chapter 1.2 the purpose of the study was mostly to create a webpage for internal usage at the company. During a typical day the customer-support received many customer errands concerning NEXT. Some of the errands entailed that customer-support contacted developers for help, to solve these issues. For all involved it became a wish that some of the simpler operations that the developers performed would be implemented into a website. This website would make it possible for customer-support to solve easier errands independently. More about how the author gathered information as such and information about the company can be read in chapter 4.1.
3 Theory

In this comparison-study two heuristic evaluation methods were reviewed. Usability evaluation was also used to validate whether the administration tool achieved high satisfaction. These terms will in the following chapter be explained, along with several other methods that were used during in the study.

To begin with the research questions will be restated:

RQ1: "Can Usability Evaluation Methods be applied in a cheap and formative manner when implementing a website, and still yield a high satisfaction and efficiency?"

RQ2: "Are Gerhardt-Pawals principles more effective than Nielsen's heuristics at yielding more usability problems of high severity rating in shorter time?"

RQ1 requires a definition of usability. When assessing whether a webpage has achieved a high user-satisfaction, how can this be measured? Chapter 3.1 gives several definitions of how usability can be defined. Chapter 3.2.5 introduces a set of metrics for recording satisfaction. Chapter 3.2.6 gives a short description about summative and formative usability evaluation and why formative was appropriate for this study. Both RQ1 and RQ2 require literature regarding UEMs. Chapter 3.5-3.7 explain different types of methods, and how they are used. Since the study incorporated guidelines like working close to the company and its employees, chapter 3.2 will present the background on why this is important.

3.1 Usability

Brooke (1996) explains that usability is a quality that is difficult to measure. He writes: "Usability does not exist in any absolute sense; it can only be defined with reference to particular contexts. This in turn means that there are no absolute measures of usability". He also summarizes "... a general quality of the appropriateness to a purpose of any particular artefact" as being a definition of usability.

Another broad definition of usability is stated as the following; “The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use” (ISO, 2010).

Another definition of usability is given by Nielsen (1993), who divides the concept into five components:

- **Learnability**: A novice user should be able to use the system with ease, so he/she rapidly can begin working.
- **Efficiency**: Once the user has learnt to use the system the productivity level should be high.
- **Memorability**: The use of the system should be easy to remember so that between breaks, the user is able to pick up the work where he/she left it.
- **Errors**: The system’s error rate should be low. If errors do occur, they should easy to recover from. Catastrophic errors must be avoided.
- **Satisfaction**: The user should be satisfied (subjectively) when using the system.

The definitions stress the importance of context. This can explain why studies like Balatsoukas et al. (2015); Chung et al. (2015); Gabriel (2007); Hundt et al. (2017) and Khajouei et al. (2017) create solutions tailored to the application domain. However, Brooke (1996) and Nielsen (1993) have acknowledged the difficulty in comparing usability in mixed domains and propose general solutions for evaluating a user interfaces. The field of such studies is called usability engineering, and the term is described as a process of producing software with the focus to meet user requirements and achieving usability (Faulkner, 2000).
3.2 The User
To achieve a good interactive system characterized by high satisfaction, one can logically assert this fact if there is a total lack of usability issues. Also, when applying the guidelines of user-centred design (UCD) interactive systems can become more usable. Additional benefits can improve productivity, enhanced user well-being, avoidance of stress, increased accessibility, and reduced risk of harm (ISO, 2010).

3.2.1 Know the user and the tasks
Incorporating the principle of UCD means in short to understand who the user is. And the minimalistic effort for developers that are implementing a system is to at least visit the customer, to get a feel for how the system could be used. This gives important insight into tasks and how the environment affects the user and what constraints are in play (Nielsen, 1993). When visiting the workplace, it will be less likely for misconceptions and mistakes later (Faulkner, 2000). This view is also held by Lekvall & Wahlin (2001) who claim that stakeholder identification is very important in the process of requirements elicitation because one gets a clearer picture of the environment in which the product will be used. Nielsen (1993) further adds that identifying the stakeholders’ tasks can also be useful, and perhaps give insight foregoing design and development process. It is also important to identify potential weaknesses. How and when does a stakeholder fail to achieve its goals? Does discomfort arise? Are resources (time/money) excessively wasted? If these questions reveal such issues, a possibility arises for a system to potentially fix them.

According to Courage & Baxter (2005) involving potential end-users in the development process may increase the general embrace of a project. Furthermore, the users may feel committed to participate, knowing that they are co-creating a product that is made for them. Some means for discovering what people are end-users in a project are interview, observation and questionnaire (Faulkner, 2000). More on this topic is covered in chapter 3.7. Finally, Gould & Lewis (1985) proposes three principles to be included in development for achieving higher usability. These are the principles of user-cantered design (UCD) and they are listed in table 2.

Table 2: Gould & Lewis (1985) list 3 important components that are involved in UCD.

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Include the user</strong></td>
<td>The designers have to locate potential users which will take part. The characteristics of the users and what tasks the users perform/will perform will be central thereafter. Also, it is very important that this is done before the design process is initiated.</td>
</tr>
<tr>
<td><strong>Using empirical measurements</strong></td>
<td>End-users should be subjected to fast prototyping for the purposes of observation, recording and analysis. The result should contain measurements of performance and reactions.</td>
</tr>
<tr>
<td><strong>An iterative design process is necessary</strong></td>
<td>An iterative cycle of design, including design, test and measurement should be implemented. Each iteration should be followed by another iteration of redesign. Many repetitions are needed.</td>
</tr>
</tbody>
</table>

3.2.2 Prototype
Gould & Lewis (1985) recommends that development should be characterized by many iterations, measurements of both prototype performance and user reactions. The definition of prototype is that a hypothesis together with a proposed design is formed to tackle a set of problems. Moreover, the best way to test a prototype is done by showing it to users/stakeholders (Pernice, 2016). Also, gathered
data from requirements elicitation should be put into such simulations to reach a successful outcome and achieving usability of a user interface (Maguire & Bevan, 2002). Furthermore, presenting users with prototypes and accepting feedback from them is encouraging for them and there are also opportunities of finding new user requirements during the process.

Prototypes can either be classed as low-fidelity or high-fidelity. Low-fidelity prototypes generally lack functionality but are cheaper to produce whereas high-fidelity prototypes are closer to the real thing and includes more interactive capabilities. The purpose of the low-fidelity type is to present the user with broad functionality and visual concepts, ranging from screen layout and design alternatives. Even paper prototypes can be used but it is not professed to be used for testing nor training. High-fidelity prototypes should in contrast provide more interactive capabilities and should be faithful in representing the end-product. High-fidelity prototypes test visual aspects of components, providing the possibility to record performance and reactions (Pernice, 2016).

3.2.3 User experience metrics

Both research questions in this study require measurements of satisfaction and efficiency. Albert & Tullis (2013) explained that a metric is a way of measuring or evaluating a phenomenon or an event. Furthermore, a system of measurement is necessary for a comparable outcome to be possible. The area of measuring user experience is simply called User Experience or more commonly referred to as UX. A set of metrics have been established in the field of UX and these are named task success, user satisfaction, errors, effort, and time (Albert & Tullis, 2013; Faulkner, 2000).

The following studies compare and evaluate different questionnaires aimed to measure user satisfaction. Stetson & Tullis (2004) conducted a study comparing a set of standardized questionnaires (SUS, QUIS, Microsoft's Product Reaction Cards and their own developed questionnaire) when evaluating a web-portal. Their findings showed that SUS was the best questionnaire. The conclusion drawn from the study was that Microsoft Word would provide the most diagnostic information and is therefore recommended for being used for helping improvement of design. Although, sus questionnaire yielded the most reliable results across the sample sizes. Moreover, sus questionnaires received a 75% accuracy score with 8 subjects, meanwhile the other questionnaires received 40-55% accuracy score.

Borsci et al. (2015) compared three questionnaires, SUS, UMUX and UMUX-LIGHT, testing them against users of varying experience. To novice users SUS was recommended, yielding the best unidimensional results. Furthermore, UMUX and UMUX-LIGHT was recommended as proxy to SUS. Bangor et al. (2008) concluded that as a metric SUS did not serve as a good standalone measurement tool. Although, combined with measurements of success rate and identified types of errors, SUS served as a good metric.

The SUS-questionnaire is industry leading and is used for its robustness and cheapness (Hvannberg et al., 2007). Furthermore, a SUS-score above 70 is said to be an acceptable score, although some reservation is added to this statement. Under 70 should imply that the user interface should be judged as bad and in need of significant improvement. Over a score of 80 is on the other hand a good score, and over 90 is an excellent score (Brooke, 1996).

A modern approach, digitalized tools like Google Analytics have been tried as a tool for usability evaluation. Hasan et al. (2009) concluded that Google Analytics is a quick way of identifying general usability issues, but that it fails to get in-depth insight into specific usability problems. It is difficult, and time consuming to interpret the data presented in the tool (a lot of data if unsure what to look for). Because some tasks are too complex to be easily compared, tools like Google Analytics help us to view data taken from tasks broken down into smaller subtasks. By further breaking down tasks into smaller
tasks, they are reduced to keystrokes and mouse-clicks (Faulkner, 2000) Worth mentioning is that metrics can be gathered from almost all usability methods (Albert & Tullis, 2013).

3.2.4 When to measure during a product’s life cycle?
The user experience can be measured during a product’s development life-cycle and according to Albert & Tullis (2013) there are two ways. These are called formative and summative usability.

3.2.4.1 Formative usability
Formative usability is characterized by the involvement of UX experts that periodically evaluate and improve the software during its life cycle. Many adjustments to improve the final product will lead to having a positive impact on a project’s ideal goal. It is also recommended to not use this principle if design changes are limited. The following questions should be answered if formative usability is applied:

What are the most significant usability issues preventing users from accomplishing their goals or resulting in inefficiency?
What aspects works well for the user?
What do users find frustrating?
What are the most common errors or mistakes made by the user?
Are improvements made from one design iteration to the next?
What usability issues can you expect to remain after the product is launched?

3.2.4.2 Summative usability
Summative usability, in contrast to formative usability, is a non-iterative process, and the evaluation is thusly performed after development has taken place.
Albert & Tullis (2013) propose that the following questions should be answered for summative evaluation:

Did we reach usability goals of the project?
What is the overall usability of the product?
How does our product compare with the competition?
Have we made improvements from one product release to the next?

3.2.5 Design processes
Faulkner (2000) proposes a set of methods for showing the end-user a representation of a design. This allows a designer to keep track and evaluate the design so far. A simulation of a user interface is one which seems to possess all functionality, but it is just a graphical representation with no real functionality tied to it (Faulkner, 2000). Four types of simulation will be briefly mentioned below.

Rapid prototyping is often good when requirements are uncertain. It is also useful when stakeholders/users are unsure about of how the system should be like, and the process allows a user to evaluate a realistic representation of the future system and give feedback (Faulkner, 2000). Wizard of Oz is a design method for testing a system without having to build it. The user is faced with something that looks like (mimics) the system, but behind the computer screen (at another location perhaps) a person acts as the system (Faulkner, 2000). Storyboards are paper-representations of an interface with interactive possibilities. A user can choose an action and try out the possibilities that are presented by the designer. If the user chooses to click on a button the designer will simulate a change to the interface mapped to that action. Paper is a cheap way of checking the design in the early stages of design, although a disadvantage is that performance should not be measured (Faulkner, 2000). Scenarios are stories about people’s interactions with a system and are used to understand if certain
tasks can be performed by using the system. Scenarios highlight the appearance and behaviour of the system and are also cheap to perform (Rosson & Carroll, 2002; Faulkner 2000).

3.3 Usability Evaluation Methods
Performing evaluation on a user interface, to measure improvement, is important when it comes to achieving high usability. In this chapter several studies about usability evaluation are presented. Gerhardt-Powals’ principles and Nielsen’s heuristics will be presented in detail.

Usability evaluation is the process of revealing issues associated with a user’s interaction with a product. Holzinger (2005) divides the methods of usability evaluation into two groups, called usability testing method (UTM) and usability inspection method (UIM). More about UIM and UTM is explained in chapter 3.4 and chapter 3.5 respectively.

When problems are found during evaluation, the next step is to improve the user interface based on these findings. According to Gould & Lewis (1985) the goal with usability evaluation is for software developers to receive feedback about a product in an iterative process. Greenberg & Buxton (2008) add that the choice evaluation method must be appropriate for the research question or problem at hand.

3.3.1 Other methods
Cognitive walkthrough is a popular method but requires an expert that knows the user very well. The expert should be able to perform a task pretending to be the user, and of course achieving results like if he/she was the actual end-user. The method is goal-based, and the background of the method is in cognitive research rather than user interface design. The advantage with this method is that it is cheap and quick, but the method is only as good as the expert (Faulkner, 2000).

Action analysis involves breaking down tasks into mouse-clicks and keystrokes. Recording the sequence of actions when conducting certain tasks gives a quantitative result, easy to compare with further tests. This method can be very time consuming and requires knowledgeable experts. Also, it is recommended that such experts have had dialogue with the end-users (Holzinger, 2005).

3.3.2 Previous studies
The studies referenced in this thesis, on the topic of usability evaluation, did not explore heuristic evaluation over several iterations. Also, they did not use the method as both a tool for steering development and design choices, and as measure of usability. Nevertheless, several components from previous studies were applied for a couple, specific, reasons.

As presented in chapter 3.4.1, the quality of performing an evaluation for heuristic evaluation partly relies on the chosen subjects/evaluators. Khajouei et al. (2017) recruited 5 evaluators for the evaluation of an Information System in Health care. These people had a background in health information technology and they received both theoretical and practical training for the evaluation methods that were employed in the study. It was concluded that the result depended on the expertise of the evaluator, and the comprehensiveness of the material presented to the evaluators.

Balatsoukas et al. (2015) uses on the other hand usability experts for heuristic evaluation. Even though chapter 3.4.1 mentions the strength of usability experts, the study did not explicitly discuss the impact that the evaluators had on evaluation. Hvannberg et al. (2007) performed a validation of heuristic evaluation with think aloud. Moreover, a comparison between Nielsen’s heuristics and Gerhardt-Powals’ principles was performed. The subjects chosen for the heuristic evaluation were university student, in computer science, divided into groups of 5. Furthermore, pre-sessional questionnaires were given for eliciting their technical experience. Sim & Gavin (2016) evaluated a Computer-assisted assessment (CAA) system with heuristic evaluation and proposed a research
question asking if novice evaluators could conduct a HE and still obtain a good result. 32 undergraduates were recruited from a course in human-computer interaction. They were also given theoretical and practical training to familiarize themselves with applying the HE. The result showed that novice evaluators could identify genuine usability issues with very little training.

Balatsoukas et al. (2015) performed heuristic evaluation in a controlled environment. Moreover, an independent researcher oversaw the evaluation and the task of constructing a single list of unique usability issues. Also, a 5-point scale was employed in the evaluation process for ranking the severity of found usability issues. 0 ("I don't agree that this a usability problem at all") to 4 ("usability catastrophe"). The three intermediate scale points represented cosmetic, minor and major. The study found that the most severe usability issues were related to statistical information. The specific heuristics linked to these findings were visibility of system status and match between the system and the real world. Furthermore, the result showed that 3 types of heuristics were never connected to usability issues. It seemed that a care pathway user-interface tended to be linked to some heuristics more than others. However, this result could possibly have been connected to the domain specific nature if the user-interface in the study.

Another study utilizing heuristic evaluation used 3 usability experts in a controlled environment. The heuristics were partly based on Nielsen’s, and partly on heuristic categories of health information system. From reviewing literature on electronic care pathways, the heuristics were improved. The process of evaluation implemented a 5-point scale for rating the severity of a problem found. The scale ranged between zero (I don't agree that this is a usability problem at all) and four (usability catastrophe). The range in between consist of cosmetic (1), minor (2), and major (3). The construction of a single set of unique usability issues was delegated to an independent researcher. The authors talked to 7 primary-care physicians that were chosen through convenience. Beforehand the participants were screened by asking them to answer an online-questionnaire. The purpose was to choose a homogenous group of people in terms of knowledge of statistics, chronic heart failure, and level of confidence in the use of web and computers. The reason for the selection was to decrease inter-subject reliability. Furthermore, the chosen subjects were invited to an evaluation session and the objective of the session/study was briefly stated. Also, consent forms were filled, and the subject was after introduced to the user-interface and the underlying functionality. No training to was given.

After the introduction, eight predetermined tasks that originated from user requirements were presented to the subjects. During a session the subject was encouraged to express his/her feelings and thoughts (talk aloud) towards the interface. A 17-inch screen laptop was used for all tests. Capturing the events on screen was achieved by a screen-recording software. The time per task and the number of faults made were recorded. Audio recording was put in place for capturing the attitude and feelings. Moreover, this was used to help the identification of the type of errors found (Balatsoukas et al., 2015).

This study used usability-testing to validate HE and to see how many predicted usability issues could be match with real usability issues found by end-users. Hvannberg et al. (2007) made the same comparison of Nielsen and Gerhardt-Powals, employed usability testing in the same fashion. The study recognized that think aloud could be employed for validating heuristic evaluation. The heuristic evaluation produced feedback concerning problems mapped to the heuristic set. Post-sessional SUS-questionnaires were then employed to measure satisfaction. Moreover, each session was timed to provide the number of unique problems found per minute. Preparatory work for the user-tests
included the development of test scenarios that were based on the results from heuristic evaluation. Scenarios were to some extent removed if they did not match the set of found predicted usability problems. The validation of the heuristic evaluation was made by comparing the number issues found by users testing (real problems) that matched with the issues found by Nielsen and Gerhardt-Powals predicted problems. If a non-match was found, the predicted issue would be called a false alarm. The study’s result showed no big difference in matching predicted problems with real problems found between Nielsen and Gerhardt-Powals. Effectiveness or efficiency did not differentiate that much either (Hvannberg et al. 2007).

3.3.3 Challenges in evaluation
According to Torrente, Prieto & Gutierrez (2012), today there are several evaluation methods, but they seldom help developers to achieve high usability. Several studies bring up numerous evaluation methods, but little help for understanding how these methods should be performed is provided to evaluators. Especially novice evaluators suffer because of this. In practice evaluators lack a clear method that describes how a UEM should be performed. This has resulted in a large variation/combination of methods because evaluators often make decisions ad-hoc, due to circumstantial and personal preferences.

As mentioned in chapter 3.1, several studies focus on developing evaluation methods for either specific application domains or more general cases. A wish in the community is to agree on what achieves high usability, and to increase the awareness of developers to the fact that even established methods are not always suitable. Thusly, many websites in production today have low usability. These are some of the current challenges in usability evaluation today (Torrente et al., 2012). The current research done on the behalf of usability evaluation focuses on the methods that produce good results at a low cost (Hvannberg et al., 2007).

3.4 Heuristic evaluation
According to Faulkner (2000) usability inspection methods (UIMs) traditionally exclude the end-user and instead give the task of performing the evaluation to a dedicated evaluator. Moreover, the “inspections” should find what is good or bad with user interface. The next step for the evaluator is to decide what to “fix”. The goal is to increase usability by removing the "bad" between one iteration and the next. Nielsen (1993) claims that UIMs are cheaper than most evaluation methods and that they can improve a user interface quickly.

According to Holzinger (2005) the most commonly practiced UIMs are heuristic evaluation, cognitive walkthrough and action analysis. But among these, heuristic evaluation is the current industry standard (Fernandez et al., 2011). Heuristic evaluation is designed to be fast and easy to use, even for novice users. There is a need for such methods since the research-area of evaluation is less covered than the research-area of user-interface design. Heuristic evaluation can be conducted in a rapid fashion saving a lot of time, which is the main purpose of such an evaluation (Guliksen & Göransson, 2002; Shneiderman & Plaisant, 2010). Nielsen (1993) claims that heuristic evaluation can detect the majority of the usability problems. found by more expensive methods. However, Holzinger (2012) explains that a disadvantage with HE is the separation from the end-user. This increases the risk for faults being missed in evaluation due to the impaired degree of exploration.
3.4.1 Perform an evaluation

Heuristic evaluation is a simple type of evaluation that is performed by an evaluator. The process of inspection involves examination and judging of whether the user interface is compliant with a set of recognized heuristics (Nielsen, 1993; Torrente et al., 2012).

According to Nielsen (1993) heuristic works good for yielding issues with few people. Usually 3-5 evaluators are the most needed. This is because the number of discovered faults rapidly increases for the first five added subjects and declines thereafter. Five evaluators are said to be enough to find 80% of the usability issues when performing a heuristic evaluation (Nielsen, 1993; Holzinger, 2012). An obvious reason for keeping the subject count low is to keep costs down (Holzinger, 2012). Holzinger (2012) adds that using non-experts does not yield as good results as when using evaluation-experts, but non-experts can be usable at times, depending on the availability of who can participate.

The next step is how to gather and filter the collected data. Connell & Hammond (1999) present a process, called problem reduction. The process results in a single list of unique problems found, tied to the heuristic evaluation with several evaluators. The first step states: Problems that are either duplicates or incomprehensible should be removed. The second step: Eliminate duplicate issues and problems of similar type between subjects, leaving a unique list of issues. Problems of the same type expressed in different ways are merged with a new issue-description. Evaluators are in the traditional sense supposed to inspect the interface alone. The inspection involves going through the interface multiple times. Moreover, they are not allowed to speak about their findings before all evaluations are completed. This is to sustain the integrity of evaluation, meaning that the data remains independent and unbiased (Holzinger, 2012).

3.4.2 Nielsen’s 10

The two methods of heuristic evaluation used in this study Nielsen’s and Gerhardt-Powals’. Nielsen’s is described firstly.

Nielsen’s heuristics are based of guidelines for graphical interfaces and preferential aspects of human characteristics. According to Nielsen (1994) this makes the set of rules good at catching a broad spectrum of usability issues.

Simple and natural dialogue. An interface should be as simple as possible since every extra component is an extra piece of information and is therefore one more thing to learn. Every additional thing on the screen is also one more thing to misunderstand and increases the time it takes for a person to grasp the interface. An interface should represent the workflow of the user in a natural way. Furthermore, colour and design are important for achieving a simple and natural way of conveying information with the user. This rule is often achieved by following the rules of Gestalt (Nielsen, 1993).

Speak the users’ language. Terminology should be used to match the users’ language. In other words, the textual dialog should not deviate, by using non-standard words, unless the words are accepted in the user-community. Also, dialog can involve the use of metaphor, mapping different icons to a users’ conceptual models. For example, the action “delete a file” could be, and is often, represented by an icon of a garbage bin. Although, a difficulty could be that a metaphor can inadvertently imply multiple meanings, lead to confusion (Nielsen, 1993).

Minimize user-memory load. Since the computer is better at remembering than us humans, this fact should be exploited. For example, the computer should generate items for the user to choose from or edit. Menus are a typical technology used for giving the user clear choices of action. In addition, a user
interface which focuses on recall should increase visibility of components of interest. Although, the drawback of giving to many components high visibility is that the visibility decreases (Nielsen, 1993).

**Consistency.** Information should be presented in the same way across screens and components should be formatted in an equal manner. If a user knows that the same actions and commands execute the same across screens, confidence while using the system will increase. Several aspects of consistency can be fulfilled by just following user-interface guidelines (Nielsen, 1993).

**Feedback.** It is important that the system has a responsive dialog and interprets data input from the user. Depending on user-preferences, novice- or expert-experience it is important to configure response-time and feedback-persistence. An example involving a printer tells us that an error message appears when the printer is out of paper. When the problem is solved, the error message on screen should automatically be removed. But other feedback might need to persist a while longer, giving the user time to acknowledge the feedback. Worth remembering is that the kind of feedback that appears is equally as important as when it does. Furthermore, the language should not be put in general terms or abstract text. In addition, it should show what has been done. Also, worth to remember is that no feedback is the worst kind of feedback (Nielsen, 1993).

**Clearly marked exits.** Users like to feel they are in control. Feeling a lack of control is something that must be minimize. Undo-commands are features to prevent getting stuck. Undo-commands are quickly learnt by the user, which means that the way the command was performed must be consistent throughout the system. Moreover, if a process takes a long period of time, over 10 seconds, the user must be given the possibility to cancel the process. The different exit mechanisms set in place must be very easy to remember (Nielsen, 1993).

**Shortcuts** are in general provided for more experienced users that aim to perform tasks quicker. Examples of such shortcuts, also called accelerators or abbreviations, are mouse double click and key commands. Key commands can be used to retrieve latest data, or data that is most relevant. Template node groups of hyper texts can be used throughout a system to provide a shortcut to different parts of the system (Nielsen, 1993).

Type-ahead and click-ahead are two types of shortcuts that give the user the opportunity of not having to wait for the computer. These shortcuts enable the user continuing with additional actions or input before a program has finished calculating a certain operation. An example of click-ahead is when a popup window appears, showing a loading bar continuously showing the progress. A user should be able to make the window disappear or at least obscure its presence. This would allow further work before the pop-up would appear again (when the loading is finished). The use of click- and type-ahead must be used with caution. When important alerts must be seen by user, these kinds of shortcuts cannot be implemented (Nielsen, 1993).

Previous commands performed by a user should be remembered by the system. Because it is often that the same input will be repeated in the future. This is good for a novice user as well as for experts. These can be implemented in search-bars, providing a list of recent search results (Nielsen, 1993).

The system can give the user default values in text areas when it is known what the user might write. Having a system that learns or gathers data about a specific user may likely save the user the time of performing repetitive commands and key-strokes (Nielsen, 1993).

**Good Error Messages.** In situations when the user is in trouble, and real harm can befall the system, it is important to give a clear representation of the situation. The representation should give the user a set of navigational options, and should follow 4 simple rules:
1. **Clearly phrased, avoiding any obscurities.**
2. **Precise rather than vague and general.**
3. **The help and options given should be constructive.**
4. **Finally, they must be polite and should not intimidate the user.**

**Prevent errors.** A system should be designed so that as few errors as possible occur. A way of achieving this is by making sections of the user-interface available or unavailable at certain points in time. By separating functionality and different actions that might interfere with each other, data can be protected (Nielsen, 1993).

**Help and documentation.** Preferably a system should be easy enough to use so that a manual is not needed. Although, when operations become too complex a supplement can become necessary. Also, documentation can be desired for users that wish to discover more about the system, perhaps learning about shortcuts for using the system more efficiently (Nielsen, 1993).

### 3.4.3 Gerhardt-Powals

Gerhardt-Powals (1996) tells us that many systems do not provide user satisfaction and the lack of such usability is often because of badly designed user interfaces. The ten principles of Gerhardt-Powals were created from the literature of cognitive science, in the context of designing human-computer interfaces. In addition, Gerhardt-Powals (1996) extracted the principles in an effort to make thousands of design related guidelines more practical for the use of designing an interface. Gerhardt-Powals (1996) produced an interface for a submarine weapon’s launch system, basing its design on the ten principles to be compared with an interface created without considering the principles (called baseline). In short, the purpose of the interface was to accommodate a submarine-operator in critical decision making and to inform about firing and own-ship problems. If you are unsure of any submarine related terms, more detailed information can be found in (Gerhardt-Powals, 1996). The principles of Gerhardt-Powals will be explained through examples from that study.

**Automate unwanted workload.** The colour red and green was used in the interface on displayed submarine targets. Showing a “red target” indicated that firing a torpedo from a certain angle was not allowed. On the other hand a green was used to indicate that the angle was valid. Using colours that humans intuitively grasp as “good” or “bad” could make a more efficient interface. The baseline-interface demanded more from the operator in terms of remembering figures and mentally checking off the requirements for validating the firing angle. This showed an apparent advantage of reducing the mental workload (Gerhardt-Powals, 1996).

**Reduce uncertainty.** Colour coding and alert messages were used for making it clear for the operator that the firing criterion had or had not been achieved. The baseline interface displayed the numbers and data in such a fashion that the operator had to personally calculate the parameters and make the decision accordingly. Having the system do these calculations and clearly presenting the operator with a yes or no response helped reduce uncertainty (Gerhardt-Powals, 1996).

**Fuse data.** This principle was applied through fusing together data pertaining to the same type of information. The interface engineered with Gerhardt-Powals took information that pertained to information about firing and grouped them together. An alert message showed a direct summary on whether the criterion for firing was satisfied. In the baseline-interface the different criterions for firing were scattered over the screen, forcing the operator to gather data mentally (Gerhardt-Powals, 1996).

**Present new information with meaningful aids to interpretation.** In both user interfaces colour-coding was used to signal “satisfied” or “not satisfied” to the operator. The cognitive-engineered
interface was clear in conveying that the own-ship criterion was achieved or unachieved by colour-coding but the understanding of why it was not achieved failed for both interfaces (Gerhardt-Powals, 1996).

**Use names that are conceptually related to function.** Labels in the cognitive-engineered interface were named so that they would inform what a certain group of alert messages had in common. "FIRING SOLUTION" was such a label, giving meaning to a set of tasks in a particular group of data related to parameters concerning firing. The baseline interface had less informative labels (Gerhardt-Powals, 1996).

**Group data in consistently, meaningful ways.** Information about the own submarine was grouped and located to the left on the screen. In addition to the grouping, speed and direction of the own-ship was located beneath. A blue, “friendly” colour, submarine picture was positioned above to enforce the fact that this information pertained to their own ship. To the right side of the screen the same information was grouped in the same way but with a red ("foe-related colour") submarine, representing a target. The grouping of information was the same across screens to achieve a consistent design (Gerhardt-Powals, 1996).

**Limit data-driven tasks.** In the cognitive-engineered interface calculations for determining distance to target and assessing firing criterions achieved became unnecessary due to colour coded alert messages, informing the operator about such matters (Gerhardt-Powals, 1996).

**Include in the displays only that information needed by the operator at a given time.** In the cognitive engineered interface, only the information about firing was displayed when the firing problem was the current issue. The same applies for the own-ship issue. Only information about own-ship was presented when the own-ship problem was the current issue (Gerhardt-Powals, 1996).

**Provide multiple coding of data when appropriate.** The cognitive-engineered interface mixed the use of high-level graphical components and numerical labels to show the same information in different ways. For example, the label saying "firing solution" would turn green if the numbers pertaining to a set of criterions were achieved. This numerical data was grouped just underneath the label, giving the operator extra information if needed. Colour-coding and labels were applied for further cognitive flexibility (Gerhardt-Powals, 1996).

**Practice judicious redundancy.** In the baseline interface redundancy was found, by placing out the same label at multiple places. Some redundancy was also found in the cognitive-engineered interface, by presenting own-ship data on both the firing and the own-ship screen. The reason for this was because such critical information was needed at all times (Gerhardt-Powals, 1996).

### 3.5 Evaluation with end-users

In this study the end-users’ views were recorded with a usability test method called think aloud. UTMs are characterized by the involvement of the end-user during evaluation, in contrast with usability inspection. A benefit with UTMs is that it provides information directly from the end-user (Holzinger, 2005). Although, while giving meaningful insight, it should be known that not all problems can be found by employing usability testing. In this study problems found with TA were referred to as real problems. There are different kinds of methods and apart from think aloud, field observation and questionnaires are the most common (Holzinger, 2005). Traditionally these methods are associated with high costs. The hiring of usability-experts and setting up laboratories for usability evaluation are some examples. Not surprisingly this make some companies hesitant of funding such methods, even if UTMs have proven to improve usability.
According to Gabriel (2007) a solution to this would be to make it possible for developers to conduct effective usability-test evaluations themselves. Designing easy to use metrics could therefore be a solution to avoid expensive expert-participation.

3.5.1 Think aloud
While performing an evaluation the user is encouraged to talk while performing a task. Tasks can be scenarios, often created from user requirements or task analysis. Tasks are created to expose the user to a simulated real-life situation. The notion “think aloud” means that the user should try to orally communicate his/her thoughts that occur during the tasks. There are different opinions about whether the immediate verbalisation may interfere with the subject's thought process (Nielsen, 1993; Faulkner, 2000).

Peute et al. (2015) compare two types of think aloud. RTA (Retrospective Think Aloud) and CTA (Concurrent Think Aloud). CTA is the standard approach of think aloud, meaning that the subject talks out loud during the test. The study used video- and audio-recording to capture reactions and attitude toward the interface. RTA proceeded like a CTA does except that the tasks are done in silence. The "think aloud" is done afterwards, while watching video and sound recordings of the tasks performed. The subject talks out loud in retrospect. Peute et al. (2015) results tell us that CTA is the more effective and efficient method. Moreover, the involvement of video-recording has been employed by several studies: Peute et al. (2015); Balatsoukas et al. (2015). The reason was to measure the efficiency of a particular user interface. The number of faults made, and the time spent on each task. Sound-recording was for gathering data pertaining to the thought-process of the subject. This helped both the analysis and identification of the type of errors made (Balatsoukas et al., 2015).

In summary, valuable information is gathered about the user-interface and about people’s effort to solve the tasks. Although, the downside to this method is that users might act differently while talking at the same time as they are performing a task. The method is also considered difficult and relies on how comfortable the user feels with the expert (Faulkner, 2000).

3.5.2 Other methods
Furthermore, field observation and questionnaires can be used as methods of usability testing. Field observation is deemed as one of the easiest test methods available. It is also cheap, and since it is designed to detect catastrophic faults that are often observed at first glance. Therefore it does not generate as much data as other test methods. Although, it should be used at final testing, not the design phase (Holzinger, 2005). Questionnaires are often used in combination with other evaluation methods to add complementary data about attitude and perception of the system.

3.6 Web programming frameworks
This this study the user interface was built as a webpage. The following present some information about the techniques the author used.

The base of a website requires two general components. These two are called front-end and back-end. According to model-driven programming front-end should take care of the graphical representation of a webpage. Data and information on the webpage should on the other hand be handled on the back-end.

There are many tools available for creating a webpage. HTML is the foundation of web development, but more code is nowadays implemented in JavaScript (React, Angular) and some in Ruby (Rails). An advantage with JavaScript are the available libraries and frameworks that provide reusability packages and functionality (Bates, 2006). The two most popular frameworks are the JavaScript-based React and
Angular. They are singled out because of the current broad usage, size of community, and performance (Forbes, 2017).

Furthermore, developing backend can be done through frameworks like Flask or Django. They are both Python-based but Flask is a light-weight framework focused on easy and quick deployment. Moreover, Django advocates the model-view-controller architecture and has the goal of being an easy way to create websites which are heavy database driven. Django it is not exclusively a back-end framework. A frontend can also be implemented, and the same is true for Flask. Both React and Angular are both exclusively front-end.

Single page application (SPA) is a new practice of building websites. The main difference between SPA and a "regular" website is that rendering should mostly be done on the client side. A SPA is characterized by the heavy use of HTTP-requests without reloading the page. Although, a disadvantage with SPA is that the client side is loaded with the task of rendering, making it difficult for devices with weaker performance (Code school, 2017).

3.7 Data collection methods

In this study several parts rely on gathering data. There are many ways to collect data (Björklund & Paulsson, 2003). Common methods are interviews, observations and questionnaires (Bell, 2016; Björklund & Paulsson, 2003).

3.7.1 Interview

The interview is a technique that consists of asking questions during a personal encounter (Courage & Baxter, 2005; Björklund & Paulsson, 2003). Data collected through interviews is called primary data, which is data obtained from an original source (Bell, 2016; Lekvall & Wahlbin, 2001). Moreover, there are three different types of interviews; unstructured, structured and semi-structured (Björklund & Paulsson, 2003; Faulkner, 2000).

The unstructured interview is a method when the questions occur during the interview and the format includes a series of open-ended questions with the goal to steer the interview towards information perceived as important (Björklund & Paulsson, 2003). Unstructured interviews are good in the beginning of the usability-engineering process, when the interviewer is unfamiliar with the stakeholders, tasks and the environment. The purpose is to capture general information about the user. It is also important to have an extra set of questions in case the interview becomes halting or if the interviewee is shy. Worth adding, the usability engineer needs to appear interested, and this can be said for all interview formats (Faulkner, 2000).

A structured interview is oppositely constructed, listing a set of alternatives for each question for the subject to pick in response (Faulkner, 2000). Characteristics of structural interviews are that questions are predetermined and listed to be asked in a specific order (Björklund & Paulsson, 2003). When specific information is required of a subject, structured interviews are preferable (Bell, 2016).

Semi-structured interview is a method in-between a structured and an unstructured interview (Bell, 2016; Faulkner, 2000). The topic of the interview is predefined, but the questions are formulated continuously after each discussion during the interview (Bell, 2016; Björklund & Paulsson, 2003). It is a flexible approach which can change between a structured and unstructured format. This method might be chosen if the interviewer wants to combine the two interview techniques to be prepared for dynamicity. An example could be, if an interviewee is unsure or nervous when given unstructured type questions, then the interviewer can switch to structure-type questions that often are perceived as less intimidating (Faulkner, 2000).

An advantage with interviewing is flexibility (Bell, 2016; Courage & Baxter, 2005). The interviewer can ask supplementary questions and note emotions (Bell, 2016). Interviews can also give rise to a deeper
understanding (Björklund & Paulsson, 2003). Although, a disadvantage could be that it takes a great deal of time (Björklund & Paulsson, 2003). Interviews are therefore not preferable if you want information from a large amount of people (Courage & Baxter, 2005).

Furthermore, Alvesson (2011) sheds light on the fact that interviewing is a complex social event and he recommends one to be fully aware of this fact to be able to critically review the data. In addition, Björklund & Paulsson (2003) also recommends that it is appropriate to avoid leading questions. Moreover, interviews are commonly used and in user-centered design. Interviews are recommended to be used continuously throughout iterations. (Courage & Baxter, 2005).

3.7.2 Observation
Another data collecting method are observations which can be done in several ways. Either by observing or participating in the activity (Björklund & Paulsson, 2003). In addition, observation can be a great complement to interviewing (Bell, 2016).

There are two kinds of observational methods, structured an unstructured observation (Lekvall & Wahlbin, 2001; Faulkner, 2000). A structured observation is when the observer knows what behaviour will occur. Often a list is constructed with the behaviours or events that are expected, and when they occur they are registered (Faulkner, 2000). Bell (2016) tells us that a structured observation has a predefined purpose. Unstructured observations are on the other hand used when no such knowledge exists. It is difficult to prepare a structured observation so that important information is not missed. Moreover, an unstructured observation is often done during a pre-study before a structured observation takes place (Faulkner, 2000).

The observation of people’s behaviour, when they perform their daily tasks, can unveil interesting aspects relatable to user-requirements. Furthermore, the analyst needs to prepare for what to look for before an observation (Faulkner, 2000). Finally, during observation one must remember the Hawthorne effect, the influence the observation itself has on the people being observed (Courage & Baxter, 2005).

3.7.3 Questionnaires
Questionnaires consist of a set of predetermined standardised questions with a set of pre-defined answers. Such answers be yes or no, or a graded scale, for example 1 to 5. It is also possible with more open-ended questions (Björklund & Paulsson, 2003).

An advantage with questionnaires is that they can record the attitude and perception towards a product. (Faulkner, 2000; Courage & Baxter, 2005). Although, a disadvantage with questionnaires is that the respondent is relatively unknown. Body language cannot be read for example, and there is room for misconception (Björklund & Paulsson, 2003). Questionnaires are also less reliable at collecting objective data and producing good questionnaires can be very time consuming (Faulkner, 2000).

3.7.4 Sampling
According to Salant & Dillman (1994) Sampling includes different methods and the purpose is to obtain information from a relatively small group of respondents to describe a larger population. The natural gain is efficiency since it takes less time and money to gather information from fewer people. Although sampling is not always necessary, because when a population is small enough the gain in efficiency is non-mentionable.
4 Method

In this study an administration tool was developed for the IT-company NEAB. A large portion of the study focused on eliciting information from company employees about what the tool should contain. This chapter explains how the tool was built, and after how it was evaluated.

The first part of this chapter focuses on the pre-study, which was conducted extensively in the hopes of finding user requirements that was not obvious from the start. Interviewing, and prototyping were not only applied for this purpose, but for the investigation of stakeholders (potential end-user). Furthermore, the pre-study was meant to keep the end-users close to the development process according UCD. Especially in the beginning of the project, when changes to the product did not cost as much as later in the product’s life-cycle.

Moreover, a large portion of time went to the literature-study and to learning of frameworks and environment-setup.

Later in the chapter the heuristic evaluation is presented, followed by the description of how the think-aloud sessions were performed.

Finally, a description on how the gathered data from each iteration was summarized and later on put in re-design.

4.1 Pre-study

Initially a relatively large pre-study was performed to get a better understanding of NEAB and potential users for the intended administration tool. The company’s need for the tool was motivated, as stated in the situational analysis in chapter 2.2.

The task of building a user interface was based on the 3 guidelines from UCD. Firstly, include the user was applied, both during the pre-study and throughout the rest of the study. During the pre-study the author learned about those who were intended to use the administration tool. However, the usage of empirical measurements were not applied as the guidelines had intended. Fast prototyping was used, more explained in chapter 4.1.4, and reactions from the users were recorded and considered during design. Although, no performance measurement took place. Thirdly, an iterative design process is necessary was clearly implemented in the study. Two heuristic evaluations (Nielsen and Gerhardt-Powals) and one usability test method (Think Aloud) was applied. The results from all evaluations were taken into consideration when redesign was initiated. Two iterations were conducted.

During the pre-study time was given for the author to learn to setup and create a website base on react, setup django and learn how react and django interacts.

4.1.1 Interviews

Interviewing was the method chosen for eliciting information about NEAB. Semi-structured interviewing chosen for this study, and as stated in chapter 3.6.1 the choice was due to flexibility and that no previous knowledge on the topic had been gathered. Interview questions were developed with non-leading questions, asking how a normal work day looked like and what people they interacted with. Questions were formed during the interview with the goal of discovering user-tasks. Chapter 3.1.2 explains why this information is important to ascertain early in a project. Also, questions were formulated to find any issues in the company's workflow, and to find a new potential end-user that could benefit from the intended administration tool. This work was necessary for the creation of a user requirements document or/and software requirements document for specifying what functionality could be expected in the end-product. It is good to make such a document, about what a product should be able to achieve on project completion. Furthermore, it was of interest to the company that
the initial work of the study consisted of research about what functionality could exist in the administration tool. Functionality in addition to what the company already had predicted. During the interview-sessions, a mobile phone was used for recording. Notes were also taken, although to a minimum because the author wanted to avoid disturbing the social quality of the discussion. This was done just in case the recorded data was lost.

The recorded data was later listened to, and resulted in a user's requirements document, LIPS (2007). The list of requirements can be viewed in table 6.

4.1.2 Literature study

Other studies and theory relevant to the thesis were reviewed via search engines (Linköping University database, Google, Google Scholar) and the library at Linköping University. The first research question, *can Usability Evaluation Methods be applied in a cheap and formative manner when implementing a website, and still yield a high satisfaction*, required some background information about usability as a definition. Literature about different evaluation methods were reviewed. Furthermore, the author attempted to link the context of usability-evaluation of websites with cheap methods and high satisfaction to the conducted research.

Keywords used throughout the literature study were: usability, usability engineering, user analysis, task analysis, heuristic evaluation, user interface, user-centred design, prototyping, usability metrics, user experience, website design principles, usability evaluation.

During the literature study, the author searched for evaluation-studies that focused on a similar application domain as in this study. This proved to be difficult since the reviewed studies employed UEMs in domains like e-commerce, games, hospital systems, school tools and web portals. The domain of the administration tool was therefore considered as rather unique. To remind the reader, the website that was produced was a simple administrative tool for handling customer data in databases connected to the company’s ERP system. Since the application domain could not be found in the reviewed studies, the focus became to find procedures that gave the best results and were independent with regard to the application domain.

The author found a study on comparing industry leading Nielsen’s heuristics with the less known principles of Gerhardt-Powals. Since it had become clear during the research that heuristic evaluation was among the cheapest of the usability evaluation methods, the author found it interesting to replicate parts of Hvannberg (2015) and compare the results. The results of that study yielded that the two methods did not differ a lot in either efficiency or effectiveness. Therefore, verifying these findings could show that Gerhardt-Powals principles are just as good at find faults as Nielsen.

4.1.3 Prototyping

A low-fidelity on-screen prototype was developed with Balsamiq mockups (2017), a rapid wireframe-tool. From the first set of interviews in the pre-study a set of user requirements had come to light. In chapter 3.3 it is recommended that for a successful user interface, gathered user requirements should be put into simulations. The author therefore chose to include a prototype as a part of confirming the already gathered user requirements, but also to discover new user requirements.

The prototype was presented individually to 4 people. Furthermore, the users were selected based on information gathered from the previous interviews (chapter 4.1.1). As mentioned, Balsamiq is a low-fidelity prototype since it does not provide any underlying functionality. The wireframe-tool made it possible to quickly assemble a mock-up and it gave a semi-realistic view of how the system might look like. By showing it to the users, feedback was possible on graphical components and general functionality represented in the prototype.
The setting in which the prototype was presented was in a conference room at NEAB. A combination of interview and think aloud was employed, and 5 users took part in separate sessions. The interview format was semi-structured and included questions about what functionality the user thought would be needed in the administration tool and why such functionality could support their work. After the interview part, the user was given the prototype to play around with the proposed concept of the support tool. The session was audio recorded with a mobile phone.

4.1.4 Programming
As presented in chapter 1.5, the author was told to use the react-framework to construct the front-end and django as the back-end component of the system. See chapter 3.6 about web programming and the frameworks of which the author based the development. Furthermore, the author chose to build a single-page-application and used django as a pure server component.

Preparatory work was done and an environment was created using webpack which could provide a node-express-server (webpack-dev-server). This was used because “hot-load” was available as a configuration option, making it possible for compilation to occur automatically after saving the code. Although, the client server was only meant for development, and was not suitable for production. Therefore, the author followed a set of tutorials that recommended the usage of webpack-loader and webpack-bundle-tracker for achieving a single-page-application, also making it more production suitable.

Quite a lot of time was spent on learning react and setting up a development environment with webpack. During the first iteration of evaluation, only the client side of the website was running, and only at the second iteration was a Django server included. Similarly, to learning the frontend aspects, it also required quite a lot of time learning Django and especially about how to incorporate Django with React.

Documentation that was helpful: React (2017) and Django (2017).

4.2 Implementation
Initially, the coding was based on the feedback from the gathered user requirements. Although, the lack of detail regarding design matters in the user requirements made decisions regarding such choices up to the author. For later iterations Nielsen, Gerhardt-Powals and think aloud provided data that steered the design-evolution of the administration tool.

In chapter 1.4 and 3.2 the importance of saving time and money was emphasized. Therefore, problems found during evaluation would be remedied according to priority. Moreover, the problems fixed would be those that had been given the highest severity rating of 3 (1-3). This guideline was not always possible to follow since the author sometimes deemed issues of 2- or 1-severity as more severe. The author chose to implement the front-end first. At this point the back-end was non-existent and server responses were represented through hardcoded responses. This was done so that the evaluation could be conducted on the user interface without an existing backend. A worry was that the front-end had to be designed with backend in mind, so that future integration would not demand large refactoring of the front-end.

4.2.1 Design guidelines
Throughout the project, the author chose to follow the basic design guidelines of Gestalt, as mentioned in chapter 3.4.2.1. These were quite simple to grasp, and they were easier to follow than accommodating and achieving all guidelines from, for example Rosson & Carroll (2002). Although some of these principles were followed, but no formal guidelines were introduced to the design process. In summary, the interface was designed after best ability, and via feedback from evaluation.
4.3 Heuristic evaluation
As described in chapter 3.4 heuristic evaluation is found to be one of the cheapest and quickest methods used for usability evaluation. Practically anyone can conduct such an evaluation and the degree of detail and structure to which the method is applied falls to the one leading the study. A study in chapter 3.3.2 compared cognitive walkthrough with heuristic evaluation. The study concluded that cognitive walkthrough yielded issues associated with learnability and heuristic evaluation found issues associated with satisfaction. They did not differ significantly when it came to the number of faults found. Choosing satisfaction as being part of the research questions stated in this study does thusly agreed with the employment of heuristic evaluation.

According to in chapter 3.3.2 combining usability inspection with a usability test could lead to better coverage of usability issues linked to a user interface. This supported the combination of heuristic evaluation and think aloud for achieving high satisfaction. Therefore, steering development in a good direction. The second research question of this study relied to some degree on the only study found on comparing Nielsen and Gerhardt-Powal. As described in chapter 3.3.2 no large difference was found between the two. Because it was the sole paper on the subject, this inspired for reproducing the comparison one more. For a comparable outcome the author decided to calculate an efficiency value, detected usability problems per minute. The effectiveness will be mentioned.

4.3.1 Choice of evaluators
As mentioned in chapter 3.4 expert evaluators yield the best results for such evaluations and the criterion is domain knowledge and background in design and evaluation. The author chose therefore to include the 5 developers at NEAB. The developers were certainly familiar with the environment as developers of both frontend and backend knowledge in the company’s own product. Although none of them had performed a heuristic evaluation before. According to chapter 3.4.1, 3-5 evaluators was recommended. An issue appeared because there were too few people for filling two groups, for each method. 5 people could not be divided into two groups. The author made the choice to bring in non-experts and several ex-students/students at the University of Linköping were asked at convenience. A criterion was that they had studied or studied a bachelor/master’s degree in a field where the use of computers/development were heavily incorporated. 5 people agreed to partake in the study. 10 evaluators were part of the usability inspection.

Reproducing the method of Hvannberg’s (2007) heuristic evaluation called for the labelling of two groups, A and B. Each group consisted of 5 people and the experts and the students were evenly distributed. The author knew that this this would impact the homogeneity of the groups, but the importance was to achieve two groups of a similar mix. The distribution was thought to decrease the risk of imposing an imbalance to the result. Group A would conduct a heuristic evaluation with the help of Gerhardt-Powals’ principles while group B would use Nielsen’s heuristics.

4.3.2 Nielsen and Gerhardt-Powal
Two lists were developed based on Nielsen’s heuristics (chapter3.8.2) and the principles of Gerhardt-Powals (chapter 3.8.3). The two original sets were interpreted and was made to make each principle short and concise. Since the literature originally was in English a translation was made to Swedish since the evaluators were Swedish. The translated lists were the ones applied in evaluation. Below in table 3 and 4 the lists are shown as they were before they were translated.
### Table 3: List of Gerhardt-Powals principles that were used during evaluation.

<table>
<thead>
<tr>
<th>Principle number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Remove mental calculation and unnecessary thinking.</td>
</tr>
<tr>
<td>2</td>
<td>Data and visual content should be displayed in a clear manner.</td>
</tr>
<tr>
<td>3</td>
<td>Bring together lower level data into a higher-level summation to reduce cognitive load.</td>
</tr>
<tr>
<td>4</td>
<td>New information should be presented within familiar frameworks (e.g., schemas, metaphors, and everyday terms) so that information is easier to absorb.</td>
</tr>
<tr>
<td>5</td>
<td>Names and labels displayed should be context-dependent, which will improve recall and recognition.</td>
</tr>
<tr>
<td>6</td>
<td>Within a screen, data should be logically grouped; across screens, it should be consistently grouped. This will decrease information search time.</td>
</tr>
<tr>
<td>7</td>
<td>Use colour and graphics, for example, to reduce the time spent assimilating raw data.</td>
</tr>
<tr>
<td>8</td>
<td>Exclude extraneous information that is not relevant to current tasks so that the user can focus attention on critical data.</td>
</tr>
<tr>
<td>9</td>
<td>The system should provide data in varying formats and/or levels of detail in order to promote cognitive flexibility and satisfy user preferences.</td>
</tr>
<tr>
<td>10</td>
<td>Principle 10 was devised to resolve the possible conflict between principle 6 and 8, that is, in order to be consistent, it is sometimes necessary to include more information that may be needed at a given time.</td>
</tr>
</tbody>
</table>

### Table 4: List of Nielsen's heuristics that were used during evaluation.

<table>
<thead>
<tr>
<th>Heuristic number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simple and natural dialog: Dialogues should not contain information which is irrelevant or rarely needed.</td>
</tr>
<tr>
<td>2</td>
<td>Speak the user's language: The dialogue should be expressed clearly in words, phrases and concepts familiar to the user.</td>
</tr>
<tr>
<td>3</td>
<td>Minimize the users' memory load: The user should not have to remember information from one part of the dialogue to another.</td>
</tr>
<tr>
<td>4</td>
<td>Consistency: Users should not have to wonder whether different words, situations, or actions mean the same thing.</td>
</tr>
<tr>
<td>5</td>
<td>Feedback: Should always keep users informed about what is going on.</td>
</tr>
<tr>
<td>6</td>
<td>Clearly marked exits: Users often choose system functions by mistake and will need clearly marked &quot;emergency exists&quot;.</td>
</tr>
<tr>
<td>7</td>
<td>Shortcuts: Accelerators- unseen by the novice user- may often speed up interaction for the expert user such that the system can cater to both inexperienced and experienced users.</td>
</tr>
<tr>
<td>8</td>
<td>Good error messages: They should be expressed in plain language, precisely indicate the problem, and constructively suggest a solution.</td>
</tr>
<tr>
<td>9</td>
<td>Prevent errors.</td>
</tr>
<tr>
<td>10</td>
<td>Help and documentation: Even though it is better if a system can be used without documentation, it may be necessary to provide help.</td>
</tr>
</tbody>
</table>
4.3.3 Evaluation session
The setting at which each evaluation was performed varied depending on the type of evaluator. If the evaluator was a NEAB employee, the session was performed at the company in a conference room. It was explained in chapter 3.4 that a calm setting is important for any evaluation. Also, the integrity of each test was kept intact to some extent since each evaluator performed the evaluation separately. The location of each session with the students was chosen after their preferences.

The evaluation then followed the following steps:

1. Inform about step 2-6 and explain the purpose of evaluation and what impact it may have on the implementation process.
2. Introduce the administration tool and its functionality. If the evaluator was not a NEAB employee, some background should be given about NEAB and the purpose of the administration tool.
3. Read the heuristics/principles together with the evaluator. Discuss any uncertainties.
4. Explain to the evaluator that he/she should inspect the user interface several times. Ask the person to try to either map perceived problems to specific heuristics/principles or pick one heuristics/principle at a time and scan the web-page for non-compliances. Encourage the use of the heuristics/principles because the evaluator might be unfamiliar to the method. Try not to interfere until the evaluator is really stuck.
5. For each reported problem a short description, together with an id, should be written into the form shown in figure 1.
6. After the evaluator has reported faults to his/her satisfaction, the evaluator should go through each issue again and give it a severity rating (1-3).

For calculating the number of faults found per minute, the session should be clogged. Starting when the evaluator begins the inspection and ends when the last severity-rating has been given out.

4.3.4 Data gathering
The gathering of data was performed using a standardized form on which comments from an evaluator was written down on during a session. The form was replicated from a study but was only partly implemented. The original form included columns with the following points:

1. A numeric identifier.
3. Difficulties that the problem inflicts on the user.
4. Specific context (the location of the problem in the interface) 5.
   Possible causes of the problem (what is wrong in the design) 6.
   Severity rate, containing 3 levels: severe, moderate and minor 7.
   Mapping to heuristics/principles.

Figure 1 illustrates how the form was implemented during a session. It was decided, since the user interface was relatively simple, that 3 and 5 of the original form was removed to save time.
4.3.5 Data filtering
The handling of data gathered from heuristic evaluation in this study incorporated the use of problem reduction (chapter 3.4.1). The method consists of narrowing down the results from reported usability issues from evaluators and creating one single list of unique problems. After that the 10 evaluations had been performed, the forms were transcribed to a OneDrive-document. Each form on OneDrive was iterated over and problems that were difficult to interpret or were duplicate were removed. The next step involved removing duplicates between subjects. A single list was obtained resulting in a single document. The final step consisted of removing problems of the same type resulting in one list of unique problems per method. The author was responsible for performing the filtering.

4.4 Think Aloud
The purpose of a think-aloud test-session was to compare predicted usability problems, revealed by heuristic evaluation, found with real problems found by the end-users using TA. Chapter Problems found during a test session would be compared with these for the validation of Nielsen’s heuristics and Gerhardt-Powals principles.

4.4.1 Choice of evaluators
The population of end-users were discovered during the pre-study and amounted to 7 people. The type of stakeholders had been identified as people that needed an administration tool were the employees of NEAB that worked close to customer-support. Because 3-5 people were recommended for evaluation and the fact that the administration tool was relatively simple, the author deemed it unnecessary to include the entire population. 5 people were then chosen whereof two were experienced in their respective roles while three were relatively new at the company with 3 months or less experience. One of the subjects had been introduced to the low-fidelity prototype, which bare
high resemblance to the high-fidelity prototype. This experience was noted but other than that the subjects had no previous training with the user interface beforehand.

4.4.2 Evaluation session

The selected participants were contacted to take part in an arranged test session in the conference room at the company. The following steps were performed during each session:

1. The users were introduced to the functionality of the administration tool and each subject was given the chance to become familiar with the interface before commencing the test.
2. A brief explanation of how the test would to be conducted was also given. The tasks (table 5) were created from data elicited in the pre-study.

- The author also tried not to intervene
- The author encourages the subject to talk aloud.
- The expert should look at problems that happen during the session instead of seeking confirmation of expected issues.

Table 5: Created tasks for scenarios.

<table>
<thead>
<tr>
<th>Task id</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><em>Create a customer database</em>: The user will attempt to create a new customer database.</td>
</tr>
<tr>
<td>2</td>
<td><em>Update modules</em>: The user will receive instructions for activation and deactivation of number of modules</td>
</tr>
<tr>
<td>3</td>
<td><em>Solve common issue 1</em>: The user will attempt to navigate the interface with the goal to fix a customer-related issue called common-issue 1.</td>
</tr>
</tbody>
</table>

For screen and audio recording a free application, SimpleScreenRecorder (2017), was used. The test was performed on a Laptop with a mouse. After each session, the user would submit a questionnaire (Brooke, 1996) called System Usability Scale, a robust way of documenting the user's satisfaction in using the interface. Before filling it out, it was encouraged to answer honestly.

4.5 Prepare new iteration

The process of the gathering data from each evaluation resulted in three lists and the next step was to choose which problems that would be fixed. As illustrated in figure 2 the process consisted of 3 stages of deciding which problems would be chosen for the implementation list for next iteration. The problems of severity 3 would be added and in the case of think-aloud, the problems that were perceived as most distressing or most frequent among the end-users would be added. Lifting problems of low- or medium-severity was optional, if the author thought that an issue had been dealt a lower rating than it should have been. Lifting low- or medium-rated issues was decided by the author. This was probably most appropriate since the author saw the entirety of the project. If a problem was deemed severe but was not feasible to remedy in the time-frame or that it was not as important as the evaluator may have thought.
Figure 2: The implementation process described 4 stages; 1. Given a list of implementations TODO, 2. Implement the TODOS, 3. Perform usability evaluation, generating a list of found issues, 4. Of these issues, choose which to fix and include in the next iteration.
5 Results

In this chapter, the result will be presented, starting with the results from the pre-study interviews and the balsamiq prototype. Secondly, the results from comparing Nielsen’s heuristics and Gerhardt-Powals principles will be shown, together with the results from TA sessions and SUS questionnaires.

5.1 Pre-study

During the pre-study interviews, a literature study, a prototype, and programming tutorials were applied. Interviews were performed to gather information about different stakeholders by asking, among other things, what a normal work day looked like. This process made the author gain knowledge about different roles in the company, and what use of the administration tool they imagined beneficial. The interviews resulted in a company description (chapter 2.1) and situational analysis.

During the pre-study information was elicited about NEAB’s current need for implementing an in-office website. Findings showed that the company needed a website giving functionality to facilitate cases when employees working with customer support needed to contact NEAB-developers to solve certain issues. A shared opinion was that the administration tool needed to be able to add customer databases without contacting the developers. Additional employees wanted functionalities that achieved a particular “standard fault message”-fix, and activation and deactivation of modules. Map integration was an example of such a module. No serious problems were found after 9 interviews with different stakeholders in the company. Of the few that were found regarded the forced communication between the support team and developers. The communication had to do with tasks that only the developers were permitted to perform. Therefore, if the support team were given a tool providing them to such tasks that they could perform by themselves it would make an improvement in the daily work of both the support team and developers. Initially in the study the author conducted interviews and most people felt irritated that “simple tasks” that were executed by a system developer in 2 seconds could be delayed for as much as a half hour before a system developer saw the memo. Also, the developers that were asked told the author that 5-6 times a day they engage in irregular tasks that come from a customer via the support team. These tasks could vary in time. According to the developers some of these issues could be mitigated to the support team if an administration tool was implemented, but some of the more complex tasks (fixing system-bugs or faults) might be too complex to include in such a tool. For safety reasons among others.

Furthermore, user requirements were elicited from this process and with help from the author’s supervisor at NEAB information about to technical restrictions, customer-data and system-requirements was gathered.

An Anja string was required for creating a new customer database. The name originated from NEAB’s own instance of NEXT called Anja which held functionality to create new customer database was created. The company’s current procedure for creating a new database required a NEAB employee to perform a sequence of actions in Anja which would generate an "Anja string". This string would then be sent to a developer who would apply the string within a script that performed database queries, ultimately creating a new customer database. In addition to the "anja string", the choice of a project type "project" or "small business" was required. It was understood that the administration tool would pass an "anja string" and give the option of choosing project type.

As mentioned in the situational analysis several interviewees expressed what issues that faced in their daily work. Problems had to do with customer errands that often demanded the help of developers. Such issues were often quickly fixed by a developer and several employees felt frustrated that tasks
that required one command in a UNIX terminal was not available for non-programmers. Interviews held with company employees gave a picture of what functionalities might be useful to implement in an administrative tool. The author added a 6th requirement because it was an aspect that might appeal customer support. Some motivation was also mentioned in company description about that company was estimated to grow 50% annually. This was ultimately not implemented in the study because of time limitations. Several of the suggested functionality were down-prioritized for this reason. The user-requirements that were finally chosen for this study were listed in table 6.

Table 6: Requirements that are required to be implemented. The five listed are taken from User Requirements document.

<table>
<thead>
<tr>
<th>Requirement id</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The interface shall be able add a new client</td>
</tr>
<tr>
<td>3</td>
<td>The interface shall be able to resolve the “fail to update cost” problem.</td>
</tr>
<tr>
<td>4</td>
<td>There shall be a possibility to activate modules.</td>
</tr>
<tr>
<td>5</td>
<td>There shall be a possibility to deactivate modules.</td>
</tr>
<tr>
<td>6</td>
<td>There shall be a possibility to activate e-Invoice.</td>
</tr>
</tbody>
</table>

When applying the Balsamiq prototype as described in chapter 4.1.3 NEAB employees responded to the general functionality of the user interface that was drawn from the user requirements (table 6). Each requirement was given an own screen apart from requirements 4 and 5 that share a page.

In separate sessions with NEAB employees, they got familiarized with the prototype. This gave the author feedback on page-design and user-requirements. Furthermore, the evaluation exposed a lack of knowledge about certain details. This helped the author to gain additional knowledge about user requirements and about design issues that the employees perceived as problematic or confusing. Comments from the session:

- “Small business, or Project, how do I choose this? … ‘Search customer’, I think of a pop-up that appears”
“You have captured the essence of it... Simplistic instead of flashy”

“All modules do not need to show, some of them are always active... Some customers have old names in the database, may be good to think of in search engine”

These comments were taken into consideration. The first and third quote completed information about in-data and affordances. The second comment gave insight into the company's strategy for internal procedures. Design should aim for simplicity.

5.2 First iteration

The first usability inspection was performed with Nielsen and Gerhardt-Powals yielding two lists of predicted issues, from the inspection of the administrative-tool by group A and B. Think aloud sessions with the end-users generated onscreen-video and audio which were analysed yielding a third list of problems. The end-users completed each session with a SUS-questionnaire, giving an average SUS-score. The high-fidelity prototype consisted of front-end and no back-end and the simulation of backend was hard-coded. All user-requirements in table 6 were implemented.

5.2.1 System description

From the first iteration of this study a low-fidelity prototype was produced. The system's different functions will be described, as well as what the different frameworks used in development.

The architecture of the prototype was simplified so that evaluation could start sooner. The system consisted solely of the client-side and no back-end. Server-responses were simulated with hard-coding on the client-side. Moreover, the client ran on a webpack-dev-server locally on a laptop.

The web-page was implemented with the possibility to navigate across four screens and the overall design was based on the balsamiq prototype (figure 3). The button-panel consisted of three buttons, located to the left as a menu that always was visible. Each button corresponded with a screen containing functionality from each user-requirement, listed in table 6.

Functionality included a search-algorithm, seen in figure 4, for Activate/deactivate modules. This web-component was chosen to be able to find and select a customer for which modules needed to be added or removed. Furthermore, "activation and deactivation" of modules were given an own page, and so was the "creation of customer databases" (Requirement ID 1). In figure 8, implementation of module selection is located to the right. The representation of whether a module was paid for by the chosen customer was a crossed box, as seen in figure 9. In a live-situation, customer support might receive a request for adding the module “Bokförda timmar”. Firstly, the specific customer would be found via the search engine. After pressing the button below the search area, modules that the customer already had chosen would receive a cross. Clicking the box to the left of the label would give that box a cross, indicating that it has been chosen. To complete and save the choices made the button called update had to be clicked.
Providing functionality for adding a customer-database fulfilled user-requirement Id 1, and the screenshot of the implemented screen can be seen in figure 7. The screen was implemented with reactstrap’s form-component. In addition to text input, checkable label-components were implemented for providing the choice of project type. From the pre-study it was learnt that a string, generated from NEAB’s own instance of NEXT, was required for creating a new customer-database. As illustrated in figure 4, a text input was implemented to take the Anja-string. The choice of the type of project was given below the text input. “Project” or “small business” was the internal notation at NEAB for specifying what type of product configuration a customer had chosen. In the bottom of the screen the author placed a button for launching the customer-database. If successful, a green message showed up, notifying that the operation succeeded. Otherwise it was red.
For achieving good feedback, the author chose to convey affirmation to the user after a customer-database was created. In addition, an extra button for adding another database was added, appearing if the previous database-creation was successful. This button would reset the form. Moreover, the project-type was set to “project” as default since employees told the author that that type was the most common.

5.2.2 Heuristic evaluation
The evaluators found it, with varying difficulty to connect issues to the list (heuristics/principles) that they were given. Some of the frustration was expressed towards the list because it did not embody the type of issues that they perceived as important. To some extent this resulted in a resistance to use the list. The tendency was partly counteracted by repeatedly encouraging the usage of the lists. Nevertheless, the heuristics became a barrier for critiquing the user interface to some extent. After additional encouragement and explaining that the product would ultimately benefit from an increased effort, very few abandoned the list.

The evaluators from NEAB contributed especially with knowledge about input-formats in relation to NEXT and user-related language. They also brought more knowledge like the “anja string” (chapter 5.1.3). An “anja string” consisted out of three parts and these had to be checked before calling the API of NEXT’s master database. The check would have to assert that the first part, the customer number, had 6 digits. The second part would be a text string, representing the customer’s name. The last assertion would check for 10 digits, representing the organization number. The other evaluators, students and ex-students, commented more heavily on visual design and general functionality.

Group A (Gerhardt-Powal’s principles) found 52 unique problems and of these 9 of them were considered severe. The discovered issues with severity rating 3, and some randomly chosen issues are listed in table 8. Each session took on average 38 minutes and yielded about one fault per every four minutes. In 20 minutes, 4.71 unique faults would be revealed during. The spread on how many issues of each severity-type is illustrated in figure 4. The distribution shows that most of problems found were of low severity and almost as many were of medium severity. During the heuristic evaluation when applying Nielsen (Group B), 45 unique problems were yielded. The discovered issues with severity
rating 3, and some randomly chosen issues are listed in Table 7. The evaluators found that out of these issues 12 were perceived as severe. The spread of found problems with different severity-rating is illustrated in Figure 7. If compared with Figure 4 the number of severe usability issues were 3 more when using Nielsen, but the total number of faults found were 8 more when using Gerhardt-Powals. Applying Gerhardt-Powals took on average 45 minutes and yielded about one fault per every four minutes. In 20 minutes 4.65 unique faults would be revealed during this time.

![Nielsen iter. 1](image)

![Gerhardt-Powals iter. 1](image)

*Figure 7: 45 problems were found from Nielsen’s evaluation. This chart shows the distribution based on severity.*

*Figure 6: 53 problems were found from Gerhardt-Powals. This chart shows the distribution based on severity.*

Table 7: Comments of the most severe problems found, and four randomly picked problems of less severity, that were recorded during expert evaluation with Nielsen’s heuristics.

<table>
<thead>
<tr>
<th>Problem id</th>
<th>Short description</th>
<th>Where</th>
<th>Severity degree</th>
<th>Principle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Current search format is not logical (dropdown expected)</td>
<td>Modules, common issues</td>
<td>Severe</td>
<td>P1, P2, P6</td>
</tr>
<tr>
<td>2</td>
<td>List customers in different ways if a user’s memory fails</td>
<td>All search bars</td>
<td>Severe</td>
<td>P9</td>
</tr>
<tr>
<td>3</td>
<td>Case sensitive in search bar</td>
<td>All search bars</td>
<td>Severe</td>
<td>P1</td>
</tr>
<tr>
<td>4</td>
<td>Which customer’s modules am i working towards?</td>
<td>Modules</td>
<td>Severe</td>
<td>P2</td>
</tr>
<tr>
<td>5</td>
<td>Reset feels like undo (Reset button does not do what it says)</td>
<td>Modules</td>
<td>Severe</td>
<td>P5</td>
</tr>
<tr>
<td>6</td>
<td>Modules (boxes) should be sorted in alphabetical order</td>
<td>Modules</td>
<td>Severe</td>
<td>P6</td>
</tr>
<tr>
<td>7</td>
<td>Feedback should be given for changed modules</td>
<td>Modules</td>
<td>Severe</td>
<td>P2</td>
</tr>
<tr>
<td>8</td>
<td>Update button should ask if certain</td>
<td>Modules</td>
<td>Severe</td>
<td>P10</td>
</tr>
<tr>
<td>9</td>
<td>No way to regret or edit created database</td>
<td>Modules</td>
<td>Severe</td>
<td>P2</td>
</tr>
<tr>
<td>10</td>
<td>Selected button/page is not clearly indicated</td>
<td>Menu buttons</td>
<td>Medium</td>
<td>P9</td>
</tr>
<tr>
<td>11</td>
<td>Boxes appear before they are needed</td>
<td>Modules</td>
<td>Medium</td>
<td>P8</td>
</tr>
<tr>
<td>12</td>
<td>Boxes look clickable</td>
<td>Modules</td>
<td>Small</td>
<td>P2</td>
</tr>
<tr>
<td>13</td>
<td>Superfluous button “get customer data”</td>
<td>Modules</td>
<td>Small</td>
<td>P1, P8</td>
</tr>
</tbody>
</table>
Table 8: Comments of the most severe problems found, and some problems of less severity, that were recorded during expert evaluation with Nielsen’s heuristics.

<table>
<thead>
<tr>
<th>Problem id</th>
<th>Short description</th>
<th>Where?</th>
<th>Severity degree</th>
<th>Principle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Case sensitive in search bar</td>
<td>Modules, common issues</td>
<td>Severe</td>
<td>P2, P3</td>
</tr>
<tr>
<td>2</td>
<td>No descriptions about issue</td>
<td>Common issues</td>
<td>Severe</td>
<td>P9</td>
</tr>
<tr>
<td>3</td>
<td>Which customer’s modules am I working towards?</td>
<td>Modules</td>
<td>Severe</td>
<td>P5, P9</td>
</tr>
<tr>
<td>4</td>
<td>Reset buttons purpose and meaning is obscure</td>
<td>Modules</td>
<td>Severe</td>
<td>P1, P2, P4</td>
</tr>
<tr>
<td>5</td>
<td>Search does not find customer number</td>
<td>All search bars</td>
<td>Severe</td>
<td>P3</td>
</tr>
<tr>
<td>6</td>
<td>Contradictive texts and input help</td>
<td>Add customer</td>
<td>Severe</td>
<td>P4</td>
</tr>
<tr>
<td>7</td>
<td>Overall info is missing about buttons and input fields</td>
<td>All pages</td>
<td>Severe</td>
<td>P10</td>
</tr>
<tr>
<td>8</td>
<td>Boxes are not cleared after update</td>
<td>Modules</td>
<td>Severe</td>
<td>P9</td>
</tr>
<tr>
<td>9</td>
<td>Clean page when new search has begun</td>
<td>Modules</td>
<td>Severe</td>
<td>P7</td>
</tr>
<tr>
<td>10</td>
<td>Option to regret in feedback after update</td>
<td>Modules, customer database</td>
<td>Severe</td>
<td>P1, P5, P6,P9</td>
</tr>
<tr>
<td>11</td>
<td>***** Is there?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Automatic clean sheet after added customer database</td>
<td>Customer database</td>
<td>Severe</td>
<td>P3</td>
</tr>
<tr>
<td>13</td>
<td>Validates weird characters in Anja string</td>
<td>Customer database</td>
<td>Medium</td>
<td>P9</td>
</tr>
<tr>
<td>14</td>
<td>Enter-button does not do what’s expected</td>
<td>All search bars</td>
<td>Medium</td>
<td>P4, P9</td>
</tr>
<tr>
<td>15</td>
<td>Grammatical error: No capital letters</td>
<td>All pages</td>
<td>Small</td>
<td>P1</td>
</tr>
<tr>
<td>16</td>
<td>Create new button’s meaning is obscure</td>
<td>Customer database</td>
<td>Small</td>
<td>P4</td>
</tr>
</tbody>
</table>

Reviewing the types of heuristics that were mapped to problems found, some heuristics were more frequent than others. For Gerhardt-Powals P1, P2, P6 and P9 were most prominent. On the other hand, Nielsen’s heuristics yielded the following principle id’s, P1, P2, and P9.

5.2.3 End user tests

Out of the 5 end-users, customer support employees, two had had their current roles for one year or more. The remaining three were novice employees (in their current roles) with a couple months of experience. The user-tests were performed with the think-aloud method, and the subjects participated with enthusiasm. Subjects were initially very verbal, but this activity subsided sometimes, after a little while, forcing the author to encourage the subjects to verbalise their feelings towards the user-interface. Apart from this interaction, the author tried not to interrupt and did not help the subjects until they were really stuck. This happened to some extent due to a couple of bugs in the user-interface, at which point the author had to intervene and explain so that the subject could continue. The author tried to refrain from leading the subject to confirm problems that the author already expected and focused on the faults that were happening during the test session.

The comments obtained from notes and from revisioning of screen- and sound-recording resulted in a list of discovered usability-issues. Some of them are listed in table9 (most frequently mentioned).

Table 9: A sample of some of the comments recorded during TA on the first iteration.

<table>
<thead>
<tr>
<th>Problem id</th>
<th>Short description</th>
<th>Where?</th>
<th>Number of people</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Case sensitive in search bars</td>
<td>Modules, common issues</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Thought that the modules would load when customer was selected. Thought that &quot;get customer data&quot; button was one click too many</td>
<td>Modules</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Bug appeared so the user couldn’t search after a previously made module update. Had to reload page.</td>
<td>Modules</td>
<td>3</td>
</tr>
</tbody>
</table>
In this study TA was used as a tool for validating the two HE-methods. If a high match between predicted-problems and real-problems was achieved it would indicate that the HE-methods found usability-issues of high value. It was of interest for the author to check the match between the combined lists of Nielsen and Gerhardt-Powals and real-problems found during TA. Since the purpose of this study also was to decide if the mixed usage of UEMs could achieve high satisfaction. The contribution of each method was of interest but also the combined usage of inspection- and test-methods.

![Pie Chart](image1.png)  
*Figure 8: Representing the match on how many real problems that were found during heuristic evaluation.*

![Pie Chart](image2.png)  
*Figure 9: Representing the match on how many real problems that were found during Gerhardt-Powals evaluation.*

![Pie Chart](image3.png)  
*Figure 10: Representing the match on how many real problems that were found during Nielsen’s evaluation.*
Figure 7 illustrates that the heuristic evaluation yielded a 56% match with real problems. Separately measuring the match percentages between real problems and predicted problems found with the help of Nielsen’s list can be viewed in figure 8 and the respective information for Gerhardt-Powals list is illustrated in figure 9. Comparing the match between real problems and the predicted problems from the heuristic evaluation revealed that the overlap was quite low for the Nielsen’s heuristics while the overlap was significantly higher for Gerhardt-Powals principles.

5.2.4 SUS-score
This study added the use of a simple questionnaire to elicit the degree of satisfaction from the end-user. After each think-aloud session a SUS-questionnaire was given to the subject. The process took very little time and it was easy to administer. The author told the subject to answer truthfully and not to regard a bad score as hurtful. The SUS-score gathered resulted in the data shown in table 10.

<table>
<thead>
<tr>
<th>Subject</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUS score</td>
<td>87.5</td>
<td>70</td>
<td>82.5</td>
<td>100</td>
<td>67.5</td>
<td>81.5</td>
</tr>
</tbody>
</table>

Table 10: SUS score gathered from after the first iterations TA-session.

5.3 Second iteration
The second iteration followed the same process as described in chapter 5.2.

Following the implementation process illustrated in figure 2 and described in chapter 4.5 a prioritization of what problems to fix to the next iteration was performed. In accordance with the stated process, several medium and small problems were. This was because the author thought they were too severe to leave them unchanged. However, some changes were made by the author that were not linked to the unique problems found during evaluation. Some deviations from the process-plan did occur.

5.3.1 System description
New to the system in this iteration was the back-end component. Furthermore, a django development-server called runserver was implemented. As illustrated in figure 11, django was setup with a sqlite3 database which simulated customer data. This was earlier hardcoded on the client-side. The set of modules were therefore possible to edit and save, increasing the lifeliness to the end-product. In addition, when the simulation of creating a customer database was performed, metadata about the customer database was saved in the database. This was not relevant for the user experience but was thought to provide meaningful meta-data for future implementation, by tracking actions and providing history to the administration tool. A customer-database (postgresql) was provided by NEAB for showing realistic customer data. In addition, the sqlite3-database was also used for saving user-profiles. New for this iteration was a login page.
The end-product would not run locally on a laptop in production. So, implementation was made to prepare for integrating with NEAB’s production-servers. This would be realized through accessing (ssh) the company’s main server, calling a script (written by the company-supervisor) acting as the API that would be used by the administration tool for accessing real customer-data.

![Figure 11: Technical architecture of the system produced in iteration 2.](image)

In this iteration, the API was not incorporated, and instead mock-data was created for the back-end. Instead of calling the script, fake data was created. Moreover, the general design regarding navigation and functionality did not alter between iterations. Apart from a login screen, see figure 12. But there was a change of the library of react-components from reactstrap to material-ui. To be able to fix the usability-issues raised in iteration 1 the author chose from creating own components or using libraries that included components the contained the wished functionality. One of these were text-fields that included a dropdown feature. The choice was based on that the author wanted to save time by using a set of react-components with already established and commonly used.

Additional changes that were made pertained to colours and styling. For example, orange was set on the button when it was selected. This was to improve navigation. Furthermore, functionality was added for resetting a screen by clicking once on an already selected menu-button. This was a response to the evaluators opinion that the create new customer-database button on the Add customer-database screen was unnecessary.

Moreover, some evaluators perceived blue as a bad contrast to black and therefore favoured a switch. In this iteration a white background was tried instead.
In the second iteration, a newly added login screen was implemented, adding a layer of security to the administration-tool. Django's authentication-package was employed to verify a profile.

A text-field-component provided by material-ui was easy to implement and provided the tools for implementing responsive and clear feedback. The author configured the components to show error messages when faulty input was given. Furthermore, material-ui provided components that were an improvement over those in iteration 1. A text-field with an included dropdown was deemed as a suitable component for a search-bar.

Feedback from iteration 1 said that the "add customer-database"-screen was unclear, lacking in feedback and did not give an option to undo choices. Therefore, iteration 2 introduced several changes. Conveying information about the format of the anja-string was placed at the top of the screen, as seen in figure 13. A customer-number should always be 6 figures. The customer-name should be able to contain any characters. An organisation-number was required to be 10 digits long. Furthermore, a design choice was made to be able to manually add customer number, name and organisation number in a clear fashion. Therefore, three additional text fields were added. Also, it was thought to improve feedback since separate feedback was given for each of the three fields. Even if an Anja-string only was given in the first field, the other fields would automatically be filled. The input was then verified accordingly, and potential feedback was given when the button at the bottom was pressed.

According to Gerhardt-Powals’ 8th principle *information should be displayed when needed*. Feedback from the evaluations performed in iteration 1 said that the user-interface was confusing because too many items were displayed on screen, and a lot of components were displayed even if they were not needed. The order of what needed to be done was therefore unclear.
Lägg till kunddatabas

Skapa kunddatabas med Anjasträng

Kundnummer
000006

Kundnamn
neab

Organisationsnummer
000198716

Skapa kunddatabas med manuell inskrivning

Kundnummer
000006

Kundnamn
neab

Organisationsnummer
000198716

Valja en projekttyp

- Small business
- Project

Lägg till

Figure 13: New screen for adding customer databases.
Two major usability-issues that were revealed in iteration 1 were the search-bar without dropdown capabilities and the case-sensitive search-algorithm (fixed in figure 14). As previously mentioned, the text-field from material-ui that included functionality for fixing both these issues was thusly used for implementing the search bar. Integrating the component into the existing code was relatively easy apart from some versioning issues. In iteration 2 the reactions form the evaluators indicated that the search-bar was overall, a large improvement. Also, principles highlighted in the results of 5.2.2 told that the evaluators found the data in the first version was sub-optimally grouped and sometimes irrelevant. Therefore, a design decision was made, for the second iteration, to make sure that data that was not needed should be hidden until it was. As seen in figure 4 & 5, the modules were hidden before the customer was chosen. When the user entered the page of Activate module, the only thing visible was the search bar. Compare with figure 3 that showed all data from the start.
Figure 15: Modules appeared after the customer was selected. Checked means that a module is chosen for that customer.
5.3.2 Heuristic evaluation

During this iteration Gerhardt-Powal’s principles found 30 unique problems and out of these, none were considered as severe usability-issues. Some of the discovered issues are listed in table 11. Each test took on average 34 minutes and yielded on average one fault per every six minutes. If a session took 20 minutes, 3.26 unique faults would be revealed during that time. The spread on how many issues of each severity type is illustrated in figure 4. The spread of different severity-rating of the problems found is shown in figure 17. The distribution shows that Gerhardt-Powals could not find any severe problems. Comparing the faults found in this iteration with those in iteration 1, 17 faults persisted. This was more than Nielsen’s heuristics that still had 12 issues that still persisted from the first iteration.

![Graph showing the distribution of severity levels for Gerhardt-Powals iteration 2.]

Figure 17: 30 problems were found from Gerhardt-Powals. This chart shows the distribution based on severity.

![Graph showing the distribution of severity levels for Nielsen iteration 2.]

Figure 16: 28 problems were found from Nielsen’s evaluation. This chart shows the distribution based on severity.

During the heuristics evaluation when applying Nielsen 28 unique problems were yielded. Some of the discovered issues are listed in table 12. The evaluators found that out of these issues 2 were perceived as severe. The spread of found problems with different severity rating is illustrated in this figure above. If compared with figure 18 the number severe problems were two more than Nielsen’s, but the total number of faults found were only two more when using Gerhardt-Powals. When applying Gerhardt-Powals’, it took on average 33 minutes and yielded, on average, one fault per every six minutes. If a session took 20 minutes, 3.39 unique faults would be revealed during this time.
Table 11: Comments of the most severe problems found, and some problems of less severity, that were recorded during expert evaluation with Gerhardt-Powals principles.

<table>
<thead>
<tr>
<th>Problem id</th>
<th>Short description</th>
<th>Where?</th>
<th>Severity degree</th>
<th>Principle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>It should be a choice whether to search for customer number or customer name</td>
<td>Modules, common faults</td>
<td>Small</td>
<td>P6</td>
</tr>
<tr>
<td>2</td>
<td>Scroll is unnecessary work</td>
<td>Overall</td>
<td>Medium</td>
<td>P1</td>
</tr>
<tr>
<td>3</td>
<td>“Click down arrow” in search field Confuses</td>
<td>Modules</td>
<td>Medium</td>
<td>P5</td>
</tr>
<tr>
<td>4</td>
<td>Regret button should not be named clear. Use icon instead!</td>
<td>Modules</td>
<td>Small</td>
<td>P2</td>
</tr>
</tbody>
</table>

Table 12: Comments of the most severe problems found, and some problems of less severity, that were recorded during expert evaluation with Nielsen’s heuristics.

<table>
<thead>
<tr>
<th>Problem id</th>
<th>Short description</th>
<th>Where?</th>
<th>Severity degree</th>
<th>Principle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Anja string must take &quot;=&quot;</td>
<td>Add customer db</td>
<td>Severe</td>
<td>P9</td>
</tr>
<tr>
<td>2</td>
<td>Page is longer than necessary</td>
<td>Overall</td>
<td>Severe</td>
<td>P1</td>
</tr>
<tr>
<td>3</td>
<td>It is not possible to regret chosen module</td>
<td>Modules</td>
<td>Medium</td>
<td>P3, P6</td>
</tr>
<tr>
<td>4</td>
<td>&quot;Utför&quot; does not fit the buttons function. Rather &quot;verkställ&quot;</td>
<td>Common faults</td>
<td>Medium</td>
<td>P4</td>
</tr>
<tr>
<td>5</td>
<td>‘Clear field’ is not clear to what its function is</td>
<td>Modules</td>
<td>Small</td>
<td>P2, P9</td>
</tr>
<tr>
<td>6</td>
<td>Changing title as feedback confusing</td>
<td>Popups</td>
<td>Small</td>
<td>P2</td>
</tr>
</tbody>
</table>

5.3.3 End user tests
The subjects were during this iteration familiar with the concept of TA. This time it took less encouragement for the subjects to perform the procedure. During this iteration the tests yielded fewer usability-issues then the previous iteration but there still existed problems that focused more on the overall design, and newly added or changed functionality. Some of the usability-issues that were found can be seen in table 13. Moreover, figure 7 illustrates that the heuristic evaluation yielded a 29% match with real problems.

Table 13: A sample of some of the comments recorded during TA on the second iteration.

<table>
<thead>
<tr>
<th>Problem id</th>
<th>Short description</th>
<th>Where?</th>
<th>Number of people</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>It was called EFH before, now it’s called e-Invoice</td>
<td>Modules</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>A lot of text</td>
<td>Add customer db</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Case sensitive for user name</td>
<td>Login</td>
<td>1</td>
</tr>
</tbody>
</table>
The match-percentages between real problems and predicted problems can be viewed in figure 8 and 9. The overlap was quite low for the Nielsen’s heuristics, but the overlap was significantly higher for Gerhardt-Powals’ principles but compared to the previous iteration the overall overlap-percentage had decreased for both. The number of problems found in this iteration were roughly halved and so did the matched overlap-percentage.
5.3.4 SUS-score

The gathered results from the sus questionnaires are listed in table 14.

Table 14: SUS-score gathered from after the second iteration’s TA- session.

<table>
<thead>
<tr>
<th>Subject</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUS score</td>
<td>95</td>
<td>65</td>
<td>92,5</td>
<td>100</td>
<td>80</td>
<td>86,5</td>
</tr>
</tbody>
</table>

From the previous iteration the SUS-score differentiated with an increase of ca 6.1 %.
6 Discussion

The results from HE differentiated a little bit from what was expected. The methods were performed in a cheap manner and there were some clear that this had an impact on the result. However, given the efficiency and effectiveness of the UEMs the results were satisfactory.

6.1 Result
6.1.1 Pre-study
The pre-study employed one of the principles of UCD, involving the users. As mentioned in chapter 3.2, involving stakeholders and end-users will increase the chances of achieving high usability in a future product. Moreover, this principle was applied in both user-requirements gathering and in the situational analysis. During prototyping and usability-evaluation, when NEAB employees tested the system, further details were found but nothing that totally contradicted the already elicited user-requirements. A total validation of the quality of user-requirements would have required studying the system in production but this was not included in the scope of the study. Although, some validation of the quality of the user requirements was obtained during the balsamiq prototyping. At this point many of the already gathered user-requirements were verified by employees that were knowledgeable about the tasks and actions that were related to what the support tool needed to accomplish. By verifying user-requirements in the pre-study, the study may have avoided expensive refactoring if requirements later were discovered to be incomplete.

Did empirical data gathered during the pre-study help the author with making good design-decisions? Apart from applying guidelines from Gestalt, the author mostly based design decisions on the comprehension of what the end-users could benefit from. Rather than relying heavily on guidelines the author got to know the end-users. Getting to know the user and gathering user-requirements was a continuous process during the length of the study.

6.1.2 Heuristic evaluation
The result obtained from heuristic evaluation showed that Nilesen’s and Gerhardt-Powals’ were not that easy for the evaluators with no previous training. This fact was revealed when some of the evaluators became frustrated or deviated from the method. No formal querying was applied to ascertain the attitude about the heuristics/principles, but it was communicated by some the evaluators. Questionnaires could have been utilized to gather such data, as done by Hvannberg et al. (2007), but it was traded for saving time.

Group A used the list of Gerhardt-Powals and yielded 53 unique usability problems. Group A consisted of 2 NEAB developers and 3 under/post-graduates. Group B on the other hand consisted of 3 NEAB developers and 2 under/post-graduates. The fact that the NEAB developers did not split even, resulted in an unbalanced distribution of evaluators with domain knowledge. This may have influenced the internal validity of the study. Despite this group B still yielded 45 unique usability problems. This was less than group A, although group B discovered slightly more severe issues.

When employing Gerhardt-Powals’, evaluators found usability-issues most linked to the grouping of data, inconsistency, lack of varying formats and levels of detail, and unnecessary memory-load on the user. On the other hand, Nielsen's heuristics found usability issues connected to ambiguous or irrelevant dialogue, user language, and errors. The user-interface had a simple design, with a menu-bar located on the left side of the screen, always visible.
The comparison of Nielsen’s and Gerhardt-Powals’ can be replicated but the sampling of subjects could be difficult to mimic. Half of this study’s subject set were students and post-graduates from various technical educations. The other half of the set consisted of developers from NEAB. The last subset had good domain knowledge, but the overall quality of the data gathered decreased since the inter-reliability between subjects had decreased. This might have compromised the data of the study because of the low number of subjects. Moreover, a high number of participants decreases the importance of inter-reliability.

Furthermore, the validity of the experiment can be questioned on two parts. Internal and external validity. The internal validity is important for proving a causal relationship. Factors affecting inspection methods are: learning effects, usability experts’ experience, subjects’ experience, information exchange among participants, and understandability of the documents (Fernandez, Insfran & Abrahao, 2013). The participants in this study’s heuristic did not receive any training. Therefore, the internal validity may have been impaired from the lack of training. Although, training sessions were removed to speed up the process and to save time. External effects on the study’s validity could be attributed to the selection of web-artefacts. In the first iteration reactstrap was applied. Several complaints were made concerning the search bar in both the Adding customer database and Add Module page. This feedback could on the other hand not be used for deciding the usability of reactstrap of components since the author was new at web programming. During implementation the author learnt more and more about web programming which meant, the author was a better web developer during the second iteration. Since problems regarding the search-bar disappeared in the second iteration it could be said that the component used from material-ui by Google achieved usability to a significant extent, but one cannot conclude that reactstrap was a lesser library of react-components.

The result from matching real problems with predicted issues gave a percentage of 40% for Nielsen and 39% for Gerhardt-Powals in Hvannberg et al. (2007). In this study, the results differed between iterations, but the first iteration yielded a 32% match for Nielsen’s and a 44% match for Gerhardt-Powals’. The difference between Nielsen and Gerhardt-Powals match could be attributed to the specific nature of the administration-tool. Although, the second iteration yielded a 17% match for Nielsen and 29% for Gerhardt-Powals. The number of usability faults were significantly lower in the second iteration. The reason for this overall decrease may have been because of the fixed issues.

An obvious improvement to the study could have been achieved if an evaluation-expert had lead the study. The author is a masters student in computer science and the education was deeply focused on many levels of programming but not a lot on human-computer interaction. The author learnt a lot during the period of this stud, both about usability evaluation but also about web-development. Obviously, the learning-curve relating to these aspects can have affected the result.

The page-design mainly revolved around the left positioned menu bar. The navigation was very simple and not many evaluators perceived navigation as a usability-issue. All pages had although several usability-issues each and both Nielsen and Gerhardt-Powals found both similar and different issues. The most prominent usability-issues were linked to Nielsen’s 1,2,9 and Gerhart-Powal’s 1,2,6. For future work these specific principles/heuristics could be compile and tested on a small webpage of similar domain.

6.1.3 SUS-score
The average SUS-score increased from iteration one and two, indicating that the user-satisfaction towards the user-interface was improved. Although an increase was observed on average, there
were individual scores that indicated that some of the users thought that the user interface had deteriorated. User D gave the user interface a score of 100 in both iterations, and this result was perceived as a bit strange. If user D was removed from the result the average SUS score would have been 76.9 in the first iteration and 83.1 respectively. This change would mean that the overall satisfaction was lower in both iterations, but the improvement between iterations was greater. 8.1% instead of 6.1%.

A previous study, Bangor et al. (2008) performed SUS during 12 iterations over a product's life-cycle, and the plotted graph (figure 12 in said paper) covering the SUS-scores pointed to a mix of progression and regression. For example, iteration 1, 2 and 3 obtained a score of 62.5, 87.5 and 72.5. The event that marked the third iteration was labelled "New equipment added". Here a correlation could be drawn. In iteration 2 of this study reactstrap was replaced with material-ui, and several new components were introduced. This could explain why some of the users responded to the new look of the user-interface, saying that it needed more work. More iterations would have given a clearer picture of what factors that impacted on the user-interface, and would also have given clearer proof of UEMs leading to higher satisfaction.
7 Conclusion

The results of this study were not entirely aligned with the results of (Hvannberg et al, 2007). Gerhardt-Powals got slightly better scores in the metrics on which this study focused, but this does not mean the Gerhard-Powals’ should be viewed as superior to Nielsen’s.

7.1 Nielsen vs Gerhardt-Powals

Hvannberg et al. (2007) was the only study that the author found on the subject of comparing Nielsen’s heuristics and Gerhardt-Powals’ principles. The conclusion that that study came to be that the two did not differentiate so much. Neither in the aspect of efficiency nor of how many problems were found by each.

In this study the results of comparing Nielsen and Gerhardt-Powals did not differ in a big way either, but there were differences worth mentioning. The spread of the severity rate for one. Gerhardt-Powal found more problems that were related to low severity. Considering that the evaluators lacked proper training the validity and efficiency of the heuristic evaluation could be questioned. Although, the deviations may have been attributed to the fact the web-page in this study was very simple, and the type of web-page was of an entirely different type from the one in Hvannberg et al. (2007). The application-domain of the administration-tool’s simple navigation and functionality could be the reason why the two usability-inspections gave the result they did, and why some heuristics and principles were overrepresented, and some were never used.

Probably the result did depend on the evaluators who partly expressed that the heuristics and principles were difficult to map to found usability-issues during an evaluation session. Although no training was given, the evaluators did uncover issues that in varying degree matched with the results from TA. Gerhard-Powals obtained a 43% match, in the first iteration. Nielsen obtained a 32% match. In addition, it is known that usability-inspections-methods and usability-test-methods yield different types of issues and that many recommend the combination of two such methods. This may explain that “matching” might not be a perfect way of validating UIMs, or that a 50% match might be a good result, given that UIMs and UTMs complete each other.

This also calls for remembering the use of generic components. If an issue is solved and the fix is implemented and made as a generic component it could be reused, and that particular usability-issues could thusly be removed from future iterations.

7.2 Interface improvement

Both the end-users and the evaluators showed enthusiasm being part of the development, watching the progress of each iteration, feeling as they were a part of the development. This helped the author to conduct the tests and evaluations with rather ease.

The empirical data gathered from TA and SUS-questionnaires showed that an improvement took place between iteration 1 and 2. The average SUS-score increased with 6.2% which was a passable result. Also, the number of unique problems found decreased (on average) from 35 to 15. The problems that persisted could be attributed to faults that arrived with new components. For example, a login screen was introduced in the second iteration, and users commented on the lack of feedback when login failed.

A reflection on the type of faults that appeared in iteration 2, was that some faults could have been avoided if a reminder was set in place in the implementation part of the process. It is a waste to record
an issue that already had been detected during the previous iteration. An example of such an issue was issue ID1 from the first iteration of TA and issue ID3 from the second iteration of TA. Both problems concerning the presence of case sensitivity. A fault that should have been avoided in iteration 2. This calls for an issues-list from previous iterations during implementation. So that the same usability issues are avoided.

Including UEMs during the implementation-phase of a products life-cycle showed how both UIMs and UTMs uncover faults and help steer development. The most helpful feedback could be divided into two parts. Firstly, the employees of NEAB that participated in both inspections and tests unveiled problems that were connected to office-language and aspects of system-requirements. Secondly, the evaluators that were students- or post-graduates gave meaningful insight that came from outside the company, which helped fix visual details. This contribution was deemed as good since this was not important for the employees at the company where the focus lay on functionality, not enhanced graphical design.
References


Miles, M (2013). *Navigation and interaction*. Freiburg, Germany. Smashing Media GmBh.


