Final thesis

Guidelines for Web Application Usability

by

David Pärsson

LIU-IDA/LITH-EX-A--10/031--SE

2010-06-04
Final Thesis

Guidelines for Web Application Usability

by

David Pärsson

LIU-IDA/LITH-EX-A--10/031--SE

2010-06-04

Supervisor: Erik Thorin, Enea AB  
Examiner: Åke Sivertun, IDA, Linköpings Universitet
Abstract

More people are connecting to the Internet, by using computers and other devices. At the same time web applications are replacing locally installed applications. This makes web application usability an interesting and important subject.

The aim of this thesis was to find a set of usability recommendations and guidelines specifically suited for web-based applications. The guidelines were derived both from studies on how web applications and regular web sites should be designed, usability studies for locally installed applications as well as more general usability and interaction design guidelines.

A prototype was created based on the list of guidelines. The prototype was then evaluated from a usability perspective, using heuristic evaluation with Nielsen’s 10 usability heuristics, to test the validity of the guidelines.

The results of the evaluation says that while the list of guidelines can be used to help creating usable web applications, following the guidelines is not alone a mean of getting rid of all usability problems.

The full list of usability guidelines can be found in Appendix A: Guidelines.
Acknowledgments

I would like to thank Enea for giving me the opportunity to do this, for welcoming me and treating me as one of them. I would especially like to thank my supervisor Erik Thorin at Enea for his help, feedback and support which made this a lot easier. I would also like to thank my examiner Åke Sivertun at IDA, Linköping University, my opponent Johan Lindqvist for valuable comments and improvement suggestions and the employees at Enea who helped me evaluate the prototype.

Last but not least, I would like to thank Stina, my love, for making me do this and believing in me even when I doubted I could make it.
# Table of Contents

1  **Introduction** ........................................................................................................... 1  
   1.1  Background ............................................................................................................. 1  
   1.2  Purpose .................................................................................................................... 1  
   1.3  Limitations .............................................................................................................. 2  
   1.4  Method ................................................................................................................... 2  
      1.4.1  Usability Guidelines Compilation ........................................................................ 3  
      1.4.2  Prototype Evaluation ........................................................................................ 3  
   1.5  Sources .................................................................................................................... 3  
   1.6  Definitions and Terminology ................................................................................... 4  

2  **Theoretical Background** ........................................................................................... 6  
   2.1  Machine to Machine Platform ................................................................................. 6  
   2.2  Introduction to Usability ........................................................................................ 6  
      2.2.1  Traditional and Web-Based Applications ............................................................ 7  
   2.3  Web Usability .......................................................................................................... 8  
      2.3.1  How the Web is Used ........................................................................................ 8  
      2.3.2  Page Layout and Content .................................................................................. 9  
      2.3.3  Navigation ........................................................................................................ 12  
      2.3.4  Clickable Objects .............................................................................................. 15  
      2.3.5  Forms ............................................................................................................... 18  
      2.3.6  Tables and Lists of Information ....................................................................... 20  
      2.3.7  Information Dashboards .................................................................................... 24  
      2.3.8  Usability Aspects of Ajax .................................................................................. 25  
   2.4  Web Site Response Times ....................................................................................... 26  
      2.4.1  Users and Response Times ................................................................................. 27  
      2.4.2  How to Lower Response Times ........................................................................ 27  
   2.5  Prototype Evaluation Method ............................................................................... 32  
      2.5.1  Heuristic Evaluation ......................................................................................... 32  
      2.5.2  Usability Heuristics ......................................................................................... 33  

3  **Creating the Prototype** .............................................................................................. 34  
   3.1  Requirements .......................................................................................................... 34  
   3.2  Design and Implementation ..................................................................................... 34  
      3.2.1  Common Layout ............................................................................................... 35  
      3.2.2  The Map View .................................................................................................. 35  
      3.2.3  The List View ................................................................................................... 37  

4  **Prototype Evaluation** ................................................................................................ 38
5 Conclusions .............................................................................................................. 42
  5.1 Method Discussion ............................................................................................... 42
  5.2 Further Research ................................................................................................. 43

Bibliography .............................................................................................................. 44

APPENDIX A: Guidelines ............................................................................................ 49

APPENDIX B: Evaluation Instructions .......................................................................... 55

APPENDIX C: Evaluation Answers ............................................................................... 57
  5.2.1 Evaluator A .................................................................................................... 57
  5.2.2 Evaluator B .................................................................................................... 60
  5.2.3 Evaluator C .................................................................................................... 63
  5.2.4 Evaluator D .................................................................................................... 65
  5.2.5 Evaluator E .................................................................................................... 68
  5.2.6 Evaluator F .................................................................................................... 70

APPENDIX D: Implementation Notes .......................................................................... 71
  Development ............................................................................................................ 71
  Frameworks ............................................................................................................ 72
1 Introduction

1.1 Background

The Internet is getting bigger, both in terms of the number of users as well as the amount of internet based services available. The spectrum of hardware connected to the Internet is also becoming broader, with big screen mobile phones and e-book readers as examples. The internet transfer rates are continuously increasing. This gives the Internet a new kind of availability and is a great opportunity for web-based applications to take on some of the tasks previously done by traditional locally installed software.

While locally installed software is run on the operating system, the web browser defines the rules for web-based applications, so these cannot work the same way as traditional software. The web-based applications must also differ from traditional web sites since the tasks the users perform generally are more complex than just gathering information. This implies that a set of usability guidelines specifically for web-based applications are needed.

Enea has got a platform for telematics and GPS services called the Machine to Machine (M2M) project. The platform consists of customizable hardware units connected to the Internet via the cellular network. These units can be connected to a wide range of machines to monitor or control them. Major parts of the user interactions in the M2M platform are web-based and because of this Enea is interested in web application usability.

1.2 Purpose

There are guidelines for good usability and interaction design for traditional web sites [1][2] and general usability [3]. Web-based alternatives to locally hosted applications are becoming more and more available and popular [4] and since usability is an important matter to Enea they wish to further investigate web application usability practices [5]. What usability recommendations are there specifically for web applications? How can the usability guidelines for locally hosted applications and traditional web sites be applied to web applications?

Furthermore Enea wishes to expand their knowledge on how to create good and usable web applications in general. Enea has had customer requests of which a part
has been to create web applications. Some of these had to be turned down, partially due to lack of experience and knowledge in this area, even though Enea has worked on products with web-based user interfaces before. Enea now wants a set of general recommendations on how to design and implement web applications, with focus on the user interface. [5] Are there any techniques and frameworks which can make front end development for web applications easier? What best practices for high performance are there? How are usable web-based user interfaces designed?

Apart from the guidelines Enea also wants a prototype of a usable web application demonstrating possible uses of Enea’s M2M platform. This prototype should demonstrate a snow plow fleet tracking system and is to be created with the guidelines above in mind. The prototype is described in greater detail in chapter 3.1 Requirements. Enea plans to use this prototype to demonstrate the M2M platform as well as to show samples of a usable web application. [5]

Enea thus wants the following out of this thesis work:

- Guidelines for usability and interaction design in web applications.
- General implementation recommendations for web applications, with focus on the user interface.
- A prototype of a web-based snow plow fleet tracking system based on the guidelines and recommendations.

### 1.3 Limitations

No techniques requiring browser plug-ins like Adobe Flash, Java or Microsoft Silverlight are covered, although the results probably can be applied using these frameworks as well.

The prototype created in this thesis work only works properly in a recent version of Firefox.

The time available for this thesis work is limited which also limits the findings.

### 1.4 Method

The thesis consists of two main parts. Gathering and compiling usability guidelines for web applications, as well as evaluating a web application prototype created in the thesis work. Since these two parts are different, different methods were used. The first part was paid the most attention in this thesis.
1.4.1 Usability Guidelines Compilation

The guidelines for usability and interaction design in web applications were derived from existing publications – mainly articles, books and websites. There are publications giving recommendations for interaction design and usability in general [3] as well as for traditional web sites [1][2]. These recommendations were applied to the web application case where possible. Additionally, existing web application usability guidelines were examined. This gave the usability guidelines for web applications.

No guidelines, applicable to the web application case according to my personal experience, were left out, with the possible exception of recommendations not examined because of the limited time spent on the thesis.

A prototype was built based on the compiled list of guidelines.

1.4.2 Prototype Evaluation

The prototype was evaluated from a usability perspective to find the credibility of the guidelines. The method chosen to use when evaluating the web application prototype was heuristic evaluation [6]. Nielsen [7] states that heuristic evaluation is the most popular usability inspection method and that it’s done having a set of evaluators examine the interface and judge its compliance with recognized usability principles (the “heuristics”). The set of evaluators can be small. Nielsen [6] recommends the use of three to five evaluators. Nielsen, being a usability consultant [4], of course states that the best result is achieved using professional evaluators [8]. However Nielsen also states that heuristic evaluation can be performed by people with little or no usability expertise [9] and that non-experts can find many usability problems [8]. Given the low budget of this thesis work, no professional evaluators were be used. Instead the evaluators were a mixed set of people with medium to high computer familiarity, employed by Enea.

The heuristic evaluation was based on Nielsen’s Ten Usability Heuristics [10], described in chapter 2.5.2 Usability Heuristics. The application of this method is further described in chapter 2.5.1 Heuristic Evaluation.

1.5 Sources

The sources used in this thesis work are mostly web sites, articles and books about usability, web site design, interface design and general human-computer interaction.

Jakob Nielsen is often cited in this work. He is a leading web usability consultant and he holds a Ph.D. in human-computer interaction from the Technical University of Denmark in Copenhagen. [4] He has however received some criticism [11] claiming
that he has failed to disclose his personal conflicts of interest in several articles, and that he neglects the aesthetic part of the process. Most of his work is however generally accepted and appreciated.

Steve Krug is also often cited. He is a writer and usability consultant with over 20 years of experience [12]. While not having the same status as Nielsen, Krug still has a lot of experience with web usability.

1.6 Definitions and Terminology

Ajax

The term Ajax originally was shorthand for Asynchronous JavaScript + XML but has come to represent a broad group of web technologies that can be used to implement communication between the web browser and server without interfering with the current state of the page. The latter is the definition used in this thesis.

End-User or User

An end-user or user is a person or group of persons who will ultimately operate a piece of software.

Home Page

The home page is first page users see when visiting web site normally.

M2M

M2M stands for Machine to Machine and is the name of a platform created by Enea. This platform makes it possible to connect to a generic machine and monitor, control or get information from it remotely via the cellular network. More details about this are explained in chapter 2.1 Machine to Machine Platform.

Web Application

A Web Application is an application accessed via a web browser over a network such as the Internet which helps end-users solve specific tasks, such as managing emails, editing text documents or monitoring a fleet of vehicles.

Web Page

A Web Page is a single document or resource of information that can be accessed through a web browser. Web applications and web sites consist of one or more web pages.

Web Site

A Web Site, sometimes referred to as a Traditional Web Site, is a collection of related web pages, images, videos or other digital assets that are accessed via a
web browser over a network such as the Internet. Web applications are normally considered to be web sites. In this thesis however, the term only refers to the web sites focusing on information rather than tasks. One example of this is a corporate web site.
2 Theoretical Background

2.1 Machine to Machine Platform
The Machine to Machine platform (M2M) is a platform developed by Enea which aids to simplify monitoring and controlling of machines. The platform consists of units with mobile network connections. These units support a wide range of connections and protocols and can be customized and extended to support many needs. Because of this the platform can be used to monitor or control a lot of different machines. [5]

Because of the connection to the mobile network, the system is for example suitable as a mobile surveillance system or other monitoring systems independent of wired network connections. One use case of the M2M platform is to monitor and control network architecture independent of its connectivity to the Internet. Another possible use is monitoring of the level, temperature and location of tanks containing liquids. [5] In this thesis work a prototype of a system monitoring a fleet of snowplows will be created.

2.2 Introduction to Usability
Bevan [13] writes that according to ISO 9241-11 the definition of usability is:

“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”

Nielsen [9] states that usability is not a single, one-dimensional property of a user interface but has multiple components and is traditionally associated with these five attributes:

- **Learnability** – The system should be easy to learn so that the user can rapidly start getting some work done with the system.
- **Efficiency** – The system should be efficient to use, so that once the user has learned the system, a high level of productivity is possible.
- **Memorability** – The system should be easy to remember, so that the casual user is able to return to the system after some period of not having used it, without having to learn everything all over again.
• **Errors** – The system should have a low error rate, so that users make few errors during the use of the system, and so that if they do make errors they can easily recover from them. Further, catastrophic errors must not occur.

• **Satisfaction** – The system should be pleasant to use, so that users are subjectively satisfied when using it; they like it.

Quesenbery [14] sums up the mean of usability with 5 E’s. She means that in order for something to be usable, the following five characteristics must be met:

• Effective
• Efficient
• Engaging
• Error Tolerant
• Easy to Learn

Usability is critical for most web sites and web applications since the competition is hard. Nielsen [4] says that if a web site is difficult to use or if users get lost on a web site, they leave. There are plenty of other web sites available and leaving is the first line of defense when users encounter a difficulty.

Krug [1] thinks that in order to achieve high usability, everything should be as self-evident as possible. He says that designers should not make users think. Krug [1] says that his wife has put her finger on the point of usability. She says:

“If something is hard to use, I just don’t use it as much.”

In order to make people use a web site or an application it must be easy to use. Usability is however not enough, because what ultimately matter to users is the content [2].

Usability is sometimes reduced to “ease of use”, but according to the definitions of usability above that is over-simplified. Usability is a wider term although ease of use affects usability. [14]

### 2.2.1 Traditional and Web-Based Applications

In 2002 Nielsen [15] wrote about functionality-oriented web-based applications which started to emerge. Although a lot have changed in this area since then a few of his points still stand. He says that once users find the application, they must understand what it does and what it can accomplish for them, as well as its general task flow and structure. This is true for all software but often more critical for web-based applications, because users have less motivation to understand advanced features in web-based applications.

At the highest level, building usable web-based applications however isn’t any different from any other type of design [16]. The goals discussed in the previous
chapter are still the same. How these goals can be achieved will be discussed in chapter 2.3 Web Usability.

2.3 Web Usability

The most critical goal of many traditional web sites is generally communication. It could be to communicate what some company is, the value the site offers over the competition and the physical world or the products or services offered. [17]

The goal of a web application is however to help the users perform one or several specific tasks. Such tasks could be to find the best price of a specific product, search the Internet for a piece of information, manage email or collaborate on a text document. Some applications have many frequent users and thus have certain requirements while other applications that are rarely used have other priorities. For example, a web application helping users manage their email is probably used more often than an application helping users buy or sell a house. Therefore the first example probably needs to help frequent users be efficient more than the latter.

This chapter will discuss web usability practices recommended for web applications, but these can (and in many cases should) also be applied to traditional web sites. A summarized list of the guidelines in this chapter can be found in Appendix A: Guidelines.

2.3.1 How the Web is Used

The number of web sites available on the Internet is huge. According to Netcraft [18] there are over 200 million active web sites¹ accessible to the Internet users and the number of pages is of course even higher, so it’s no wonder most users are in a hurry.

As many as 79-80% of the web users scan rather than read [19] and only 16% read every word [20]. Because of this both Nielsen [2], Krug [1] thinks that one of the most important usability guidelines is to keep texts short and scannable.

According to Krug [1] the fact that users are in a hurry also affect their ability to choose. Krug says that most users “satisfice” – a cross between satisfying and sufficing – by choosing the first reasonable option they find that might suit their needs.

¹ Netcraft actually counted the number of hostnames having web servers responding to requests.
Both Nielsen [2] and Krug [1] finally says that users rarely take time to read instructions but rather forge ahead and try to accomplish what they want. These related facts, all boiling down to that users are in a hurry, are important when designing for the web.

### 2.3.2 Page Layout and Content

Most web sites consist of more than one web page and while a consistent page design is recommended [1][2] different pages have different requirements.

There are often parts of a web page which are more important than others. One might be tempted to use color or animation to highlight certain content, but Nielsen [2] warns that many users have developed what he calls banner blindness. Users know that colorful or animated areas often are banners and such areas are therefore often ignored. Krug [1] instead suggests using a clear visual hierarchy that shows users what’s important and what’s related. He gives three examples:

- **Things that are more important should be more prominent** – For example, the most important headings are the largest, have more white space or are placed closer to the top of the page.
- **Things that are related should look related** – Related things could be under the same heading, in the same visual style or in a clearly defined area.
- **Things that are part of something are visually nested** – For example, a section heading would appear above the title of a specific item, visually encompassing the whole content area of the page, because the item is part of the section.

Conventions generally exist because they work and should be followed when possible [1]. One convention that’s important to follow is to have a vertical layout [21], because users dislike horizontal scrolling [4].

### Writing for the Web

As stated above, most users scan rather than read all text on web pages. This of course affects how text on web pages should be written and laid out.

Nielsen [22] has a few recommendations for web writing:

- **Highlight keywords** – Hypertext links is one form of highlighting, typeface variations and color are others.
- **Meaningful sub-headings** – Having “clever” sub-headings instead of meaningful ones can reduce usability.
- **Bulleted lists**
• **One idea** per paragraph – Users will skip any additional ideas if they are not caught by the first few words in the paragraph

• **The inverted pyramid** style – Start with the conclusion

• **Use half the word count** (or less) than conventional writing

According to Nielsen [22], keeping text concise alone gives a higher usability improvement than having a scannable text layout. Combining the two however gives the best result. Krug [1] also notes the importance of keeping texts short and concise.

**The Home Page of Web Sites**

The home page is the first page users see when they enter a web site. The home page of a traditional web site has to communicate a lot of things. A few of them are:

• **Site identity and mission** – Instantly tell users what the site is, what it’s for and if possible why it’s better than the competition [1].

• **Site hierarchy** – The home page must give an overview of what the site has to offer, both in terms of content and features [1][2]. A well designed navigation interface can communicate this.

• **Search** – According to Nielsen [2] about half of the web users are search-dominant, meaning that they will usually go straight for the search box when they enter a web site. Therefore most home pages need a prominently displayed search box.

• **Content and feature promotion** – It should spotlighting the newest, best or most popular content or features [1].

• **Updated content** – To make users come back often or even just to show users that the web site is active the home page should contain parts that sometimes are updated [1].

• **Short-cuts** – The most frequent requested pieces of content may deserve their own links on the home page [1].

• **Registration and sign in** – If the site uses registration, the home page needs links for new users to register and old users to sign in. It also needs a way to let users know that they’re signed in. [1]

• **Show users where to start** – In order to help new users get started using the site. [1]

• **Establish credibility and trust** – For some visitors, the home page will be the only chance the site gets to create a good impression. [1]

If a web site only has one or a few important features they can be put directly on the home page for easier access [2]. If the home page on the other hand should highlight several features it may be a better idea to use one big promotion that changes over time rather than several small promotions [1]. This promotion may change each time users visit the home page or rotate on the page after a given time.

Nielsen [2] says that splash screens must die. Splash screens generally fill no purpose and only slow users down on their way to the content of the web site. One exception
to this rule is web sites that have explicit content. Such sites may warn their users before allowing them to enter.

Nielsen states that the home page may feature a larger, more prominent logotype. Krug [1] adds that the navigation of the home page may differ from the navigation on the rest of the pages, but not that much.

The home page should not be confused with the first page users see when they are logged in to a web application. Some of the rules still apply to that page, but not all. It’s not uncommon that the first page shown to users when logged in is an information dashboard [21]. These are described in chapter 2.3.7 Information Dashboards.

Credibility
The Stanford Persuasive Technology Lab has a web credibility project. They have conducted investigations on the factors affecting the perceived credibility of web pages and how the credibility of web pages can be improved. Their guidelines [23] are the following:

- Make it easy to verify the accuracy of the information by providing citations, references and source material for the information on the site.
- Show that there’s a real organization behind the site by for example listing the organizations physical address.
- Highlight expertise to make it clear if the organization comprises acknowledged experts.
- Show that there are honest and trustworthy people behind the site.
- Make contact information clear.
- Design the site so it looks professional or appropriate for the purpose.
- Make the site easy to use, and useful.
- Update the sites content often.
- Avoid advertisements, especially pop-up ads.
- Avoid errors of all types, no matter how small they seem.

Icons, Colors and Culture
Use of icons and other illustrative elements can help users understand an interface or support user recognition. If poorly executed it can however irritate, confuse or even insult users. For example, red is not a warning color in China and thumbs up is a terrible insult in Turkey. It is therefore important that designers know the users’ needs and expectations. [3]
2.3.3 Navigation

One important, if not the most important part of web site design is the navigation. It defines how end-users move around the web site, but the navigation interface also serve other purposes. According to Nielsen [2] navigation interfaces need to help users understand three things:

- **Where they are** – In order for users to be able to know where they are, all pages should indicate the user’s current location. Both on which site and where in the site’s structure.
- **Where they can go** – By providing the users with navigation options and other links, they can get a sense of where they can go. If users can acquire some understanding of the site’s structure they may also get a general idea of other places to go.
- **Where they have been** – Changing the color of links to visited pages helps or prevents users going to pages which they have already visited.

Nielsen also says that the first one – telling users where they are – probably is the most important.

Although Krug [1] uses different words than Nielsen, he agrees with on the main purposes of navigation. He also thinks that navigation has a few additional easily overlooked purposes. According to Krug the purposes of navigation interfaces are the following:

- Help users **find what they are looking for**
- Tell users **where they are**
- Tell users **what is on the site** – By making hierarchy visible, navigation tells users what the site contains.
- Tell users **how to use the site** – Good navigation implicitly tells users where to begin and what their options are.
- Give users **confidence in the site** – Clear, well-thought-out navigation is one of the best opportunities a site has to create a good impression.

How and why this should be done is explained in the rest of this chapter. A few topics closely related to navigation such as searching will also be examined.

**Show Current Location**

The web is designed in such a way that it is possible to navigate freely around the web, from virtually any page to another. Because of this the site has to be identified on all of its pages, or else users might lose track of where they are [1][2]. This is generally done by placing the site’s logo in the upper-left corner (if the page is in a language that reads left to right) [2]. This logo should be linking to the home page on all pages, except the home page itself [1]. In web applications the logo may link to the first page users see when they’re logged in [21].
But when users know on what site they are, they also need to know where they are on that site. Nielsen [2] states that if users don’t understand where on the site they are, they will not understand the site’s structure. Nielsen [2] recommends that pages should have a clear main headline and a HTML page title describing the page. Krug [1] adds that this headline should match or be closely related to the text of the link to the page.

A common convention that both Krug [1] and Nielsen [2] recommend is to highlight the area in the navigation interface where the current page is located. This can for example be done using bold text or a separate background or foreground color.

Breadcrumb navigation, as seen in Figure 1 below, is useful for showing the users location in hierarchical structures and helps users in navigating such sites [2]. A few studies have however showed that users rarely use breadcrumbs to navigate and that breadcrumbs gave no difference in task completion times unless the users were instructed on how to use them [20]. Nielsen [2] points out that generally nobody is sufficiently devoted to go through a special training to use a general web site, but he still recommends the use of breadcrumb navigation on hierarchical sites.

![Figure 1. An example of breadcrumb navigation.](Image)

**You are here:**  [Home](#)  >  [Pets](#)  >  [Dogs](#)  >  [Cute Puppies](#)

---

**Show Navigation Options**

Both Nielsen [2] and Krug [1] agree on that when pages provide navigation options, users can get a sense of where they can go. If users can acquire some understanding of the sites structure they may also get a general idea what the site contains, other than what they currently see.

To prevent users from feeling lost, pages should contain navigation [1][20] and the navigation should be persistent – placed on the same place and work the same way – on these pages [1][2]. These rules should be followed on all pages with two possible exceptions [1]. The first one being the home page which may have navigation that differs from the rest of the pages, as long as it’s not too different [1]. The second exception is pages where forms are the main actions, on which the persistent navigation might be an unnecessary distraction [1]. Krug [1] however recommends that pages with forms still should have a way to get to the home page.

As said above, the navigation interface should be displayed on (almost) every page, but there is more to a good navigation interface. The navigation interface also needs to show all the available alternatives at the same time [2]. Nielsen [2] gives an example of bad practice where the users have to move the mouse pointers over every navigation option to see what their options are.

Nielsen [2] states that users do not like it when navigation interfaces are drastically different from the ones they have come to expect from the majority of other sites.
Having the primary navigation menu to the left is preferable [20], if not using tabs which are better placed on top of the page [1]. Primary navigation should be located in a highly noticeable place of the page [17].

**Tabs as Navigation**

Tabs are widely used on many web sites as a part of navigation and Krug [1] states that the following things make properly designed tabs a good navigation option:

- Tabs are **self-evident** – It’s easy for users to understand what they do because they match a real world thing.
- Tabs are **hard to miss** – They are visually distinctive and are therefore hard to miss.
- Tabs suggest a **physical space** – Tabs create the illusion that the active tab physically moves to the front.

Krug [1] however also notes that horizontal tabs have a limited scalability.

**Searching**

According to Nielsen [2] about half of the web users are search-dominant, meaning that they use searching as their primary navigation method. Search functionality is needed if the site has many pages, and should be shown on every page [4]. This gives users a way to continue if they feel lost.

Search boxes are common enough for most users to know what they are and how they work. When providing a way to search, both Nielsen [2] and Krug [1] therefore recommends adding nothing more than a text box, a button and the text “Search”. Either the text “Search” could be added as a label or as the text of the search button. If the “Search” text is a label, they both recommend using “Go” as the button text. A common location for search boxes is at the top right corner of the page [21]. Nielsen [4] recommends that search input fields are 27 characters wide.

![Simple search box layout.](image)

A new designated search bar look has become quite popular recently. It has originated from software produced by Apple and is a text box with rounded corners and a magnifying glass to the left. [21] See Figure 3 below.

![Apple style search box.](image)

The search functionality should allow simple searches with few words and should also be case insensitive [20]. Boolean expressions in searching should in most cases
be avoided since a big majority of users does not know how to use it correctly [2]. Most important is however that the results of the user searches provide the precise information being sought and in a format that matches users’ expectations [20]. A study has shown that users heavily rely on the order of the search results, by much more often clicking the first result than the following [24].

**Browser History**

Nielsen [25] states that it is very important not to break the browsers Back button. The Back button should be one way out if users go to the wrong place. According to Krug [1], the Back button is the most used feature of web browsers. Nielsen [25] lists a few design sins that should be avoided because they do this:

- Opening a **new browser window**
- Using an **immediate HTML redirect** – Every time the user clicks the Back button the browser returns to a page that bounces the user forward again.

However, since that list was written, Ajax has become more and more popular and Ajax sites may break the Back button as well as the ability to bookmark the correct contents [26][27]. This problem can be solved by using anchors and a JavaScript library [28].

**URLs**

Nielsen [4] has a few guidelines for URLs. He states that a usable site requires:

- A **domain name** that is easy to remember and easy to spell
- **Short URLs**
- **Easy-to-type URLs**
- URLs that visualize the **site structure**
- URLs that are “**hackable**” to allow users to move to higher levels of the information architecture by hacking off the end of the URL
- **Persistent URLs** that don’t change

**2.3.4 Clickable Objects**

Navigating on web sites is generally done by clicking on hypertext links. It is therefore important to make the usage of these links as easy as possible.

**Hyperlinks**

It’s a well known fact that most users scan rather than read all text on web pages [2][1][20]. Links should therefore be styled in a way so that they catch the users’ attention and because of this it’s important that link texts are descriptive. According
to Nielsen [2] the oldest web design rule is to avoid using “click here” as the anchor text for hypertext links. Consider the two examples in Figure 4 below:

For more information about our cute puppies, **click here**.

We have more information about our cute puppies.

Figure 4. Descriptive anchor text for hyperlinks.

To this rule Nielsen [4] adds that link names should begin with the most important keyword.

The default styles for hyperlinks pointing to unvisited pages have since long been blue and underlined [21]. Nielsen [2] states that shades of blue provide the strongest signal for links, but also that other colors work almost as well [29]. Nielsen [29] also strongly recommends that underlining is reserved for links. In the same way he recommends that colored text, especially blue, should be avoided unless the text is a link.

Hyperlinks pointing to a page which users already have visited should have a duller and less vivid color than unvisited hyperlinks [29]. Nielsen [30] however states that there is one exception to this: Command-oriented functionality. Showing visited areas is unnecessary for applications in which people might want to repeat actions multiple times. If however users are going to other areas, especially to get content, they might only need one visit so showing a visited link is here appropriate.

**Buttons and Navigation**

There are a few recommendations for buttons and clickable objects to adhere to in order to ensure usability. First of all, make sure that objects that are clickable look clickable. In the same way, objects that are not clickable should not look clickable. [1][2] This means that if a link looks like a button the whole button should be clickable, not only the text. See Figure 5 for example.

![Clickable area of buttons.](image)

It’s obvious that if clickable objects are too small they are hard to click, so it’s recommended to have appropriately sized objects and clickable areas in order to increase the efficiency [21]. This is supported by Fitts’s Law [31] which predicts that the time required to rapidly move to a target area is a function of the distance to and the size of the target. Fitts’s law is among other things used to model the act of pointing to an object on a computer using a pointing device. 37signals [32]
recommends that padding is added to small links and important links such as navigational links so that they become easier to click.

The concept of double clicking is widely used in operating systems [21] and therefore it’s recommended to ensure that double-clicking will not cause problems [20].

**Visual Feedback**

Visual feedback is commonly used in computer software. The common purpose of visual feedback is to tell users that something is happening. It could be that the button the user clicked really was pressed or that the software is loading. [21]

It’s quite common for buttons and other clickable objects to have a special visual style which only is shown when users click on the objects. These styles usually simulate a physical pressed state of the object, by modifying the color to a more shaded look or slightly moving the text downwards. [21]

**Primary and Secondary Actions**

It’s not unusual to have different actions on a web page. One example of this would be to submit a form or to save its contents for later. Another is to advance or go back in a web form split up over multiple pages. The advance action enables the completion of the form and would hence be the primary action, and go back would be the secondary.

Wroblewski [33] states that by reducing the visual prominence of the secondary actions, the risk for potential errors can be minimized and users are further directed towards a successful outcome. Figure 6 below shows an example of visual distinction between the primary and secondary actions.

![Figure 6. Prominence of clickable objects.](#)

Wroblewski [33] conducted an eye-tracking study where he compared differences in the final actions of a form. He compared different alignments for actions and visual distinctions between the primary and secondary actions. The form used in the study was a left aligned form with field labels above the fields.

One of Wroblewski’s findings was that the tested visual distinction did not decrease the average completion time for filling out the forms, even if the actions were well placed. The best performance was in this case achieved when the actions were left aligned, with the primary action to the far left. He however also found that the distinction helped the users make good choices when choosing between the primary and secondary actions and that the users appreciated this distinction. He argues that
it sometimes may be a good idea to slow users down to make sure they make the right decision.

### 2.3.5 Forms

Forms are used to get data from users and they usually consist of one or more form controls with labels.

#### Form Controls

The HTML standard [34] defines several control types. It’s important to use the right control type for the tasks [4]. For example, when choosing between checkboxes and radio buttons, the latter should only be used when exactly one option in the set of options should be selected [34].

In an eye-tracking study Penzo [35] found that drop-down list boxes are very eye-catching, even more prominent than the submit button. He therefore recommends that they only should be used for important data or, when used for less important data, placed well below more important form controls.

Radio buttons are both better than and often preferred over both drop-down list boxes and open list boxes when selecting from a list of mutually exclusive items. Check boxes are better than open list boxes when selecting from a list of non-mutually exclusive items. [34]

Regular text input fields should have a size appropriate to the data asked for. For example, an input field for a five-digit zip code should not be 40 characters in length as this sets an incorrect expectation for users. [36]

Finally, the reset buttons can be rather unforgiving, resetting all controls to their initial values. This is only usable in rare cases where users enter different data in the same form many times and should therefore only be used if really needed [2][33].

#### Labels

The label HTML elements are used to attach information to form controls [34]. The labels generally contain a descriptive title of what input is wanted in the related form control [21].

There are a few good options on where to place the labels in relation to their form controls. Penzo [35] has conducted an eye tracking study where he compared different label positions, alignments and styles. His conclusions are the following:
• **Label position** – In most cases it works better to place a label above the form control instead of to the left of it. Visual separation between the label for the next form control and the previous form control is important.

• **Label alignment** – If a label is placed to the left of a form control it is in most cases best to have the label right-aligned. Left-alignment is preferred in all other cases.

• **Bold labels** – Reading bold labels is a little bit more difficult for users. Penzo however says that the test pages’ simple design may have modified the impact of bold labels.

• **Labels for drop-down list boxes** – The default value of a drop down list box may be used as a label instead of a regular label.

Labels should be placed close to their associated form controls [20]. They should be programmatically associated to form controls, for example using the `for` attribute. When associated, a label element passes focus to the associated control when the label receives focus. [34] See Figure 7 below for a good example of a form with the labels above the form controls.

![Figure 7. Form control label placements.](image)

Having associated labels is extra important for checkboxes and radio buttons, which otherwise generally have small clickable areas. In these cases, the associated labels are generally placed directly to the right of their form controls. [20][21]

**Help the Users**

There are several ways to help users in their form interaction. Below follows a list of guidelines:

• **Show what’s required** – Clearly show the form controls which require data entry. Placing an asterisks or the word “required” near the label or using a bold label are common ways to indicate this. [20]
• **Prevent errors** – Be flexible in the way users can enter data. For example, allow spaces or dashes in long number sequences. Show hints on how data should be entered if there are requirements. [37][38]

• **Detect errors** – Don’t expect that users always will make correct entries [20]. Client side validation may be used for faster feedback but server side validation is required since the user’s browser might not support client-side scripts [21].

• **Keep entered data** – Don’t make users enter the same data more than once [4].

• **Give default values** – If possible, give sane default values. This can speed up interaction and direct novice users toward a safe or common outcome. [4]

• **Focus form control** – The first form control of the page can be automatically focused. When implemented users do not have to manually select the first form element to begin inserting data [20].

• **Vertical layout** – Users doesn’t like horizontal scrolling [4]. Therefore long forms should have a vertical layout. A common convention is to have form controls in a column, left-aligned [21]. When form controls are left aligned it’s best to also have action buttons, like submit, left aligned with the most important action to the far left [33].

• **Show progress** – When forms are split up on multiple pages, show progress and how much is left [21].

• **Automatic tabbing** – If possible, providing auto-tabbing can significantly reduce data entry times for frequent users by not requiring them to manually tab from field to field [20].

• **Hide navigation** – On pages where filling out a form is the main action, the main site navigation may sometimes be an unnecessary distraction and can in these cases be hidden. [1]

### 2.3.6 Tables and Lists of Information

Tables can be used to structure data and present it to users. Lammi [39] has compiled a quite comprehensive list of guidelines for tables of data:

• **Reduce and compress data**
  
  o **Progressive disclosure** – Show only what users really have to see and give users ability to see more details. Details can be shown by using tooltips or making the content expandable to maintain the users’ flow.

  o Have an **appropriate level of precision** for numeric data. Fewer decimals make the numbers easier to scan and understand.

  o **Summarize data** when possible.

• **Reduce visual noise** – Only keep the necessary interface elements to prevent distraction. Keep those that support the structure, organization and readability of the table. Use the minimum amount of visual weight needed by for example using low contrast colors for backgrounds and thin borders.
• **Highlight what’s important** – Use a more visually distinctive styling for information that’s always important, such as totals, and information that is important at the moment, such as exceptions. However, don’t emphasize too much because then nothing will stand out.

• **Consider allowing customization** – Allowing users to select what information to show in the table can be good for frequent users.

• **Use the proper alignment** – Left align text and right align numbers. Text may be center aligned if they show a status or something else that should stand out.

The two figures below show the differences when some of the guidelines have been applied.

![Figure 8](image1.png)  
**Figure 8. A table with some visual noise and no highlighting.**

![Figure 9](image2.png)  
**Figure 9. A table with highlighting and little other visual noise.**

The list by Lammi [39] continues with a few guidelines specifically for large tables:

• **Make table headers float** – If the table becomes so large that users may have to scroll, make the table headers stay visible as the users scroll up and down the page. This adds more value for every column added. There are JavaScript libraries that can provide this functionality [40].

• **Pagination or endless scrolling** – Pagination or endless scrolling can be used to split up data that does not comfortably fit within one page.
• **Sorting** – Provide a logical sort order by default. When applicable, let users sort the table by clicking table headers. Use a link color, underline or sort icons to show that column headers are sortable. Sort order is reversed by clicking the same header twice [21]. Sorting may be done server-side but there are JavaScript libraries which can provide client-side sorting too [41].

• **Provide filtering options** – Filtering is useful when items can be classified and the users might have specific needs or for comparison. This is described in more detail in the section *Filtering* below.

**Pagination**

Pagination may be needed in order to limit the amount of information sent to the client and displayed to the user. Before adding pagination one should consider if it’s really needed [42]. One alternative to pagination is to simply have a long list which optionally is loaded with more content as the users scroll down [42]. That technique is known as endless scrolling.

Nielsen [2] states that if a list of items is being paginated it may be a better idea to group the items by a property such as alphabetically by first letter (or letters) instead of by page number. An option for chronologically ordered items is to use the group by periods of time [21].

Ateş [43] has compiled a list of good practices for pagination:

1. **Provide large clickable areas** – This rule applies to all navigational links, but is extra important to note for pagination links because of two reasons. Firstly, pagination is often omitted from the design phase and secondly, pagination obviously often consists of clickable numbers which by themselves are small. A rule of thumb is “twice as wide as the number and 1.5 times its height”.

2. **Don’t use underlines** – Underlines for links are generally a good idea but for pagination links underlines are superfluous. People know that page numbers on the web are clickable or they wouldn’t be there.

3. **Identify the current page** – Make the current page immediately identifiable for the user, significantly different from the other pages. The current page link should not be clickable and should not have any kind of visual hover-state.

4. **Space out page links** – Provide enough space between each page link so that users don’t accidentally click the wrong page number.

5. **Provide Previous and Next links** – Previous and Next links are useful in virtually any context where pagination exists, so they should be provided. They should however be clearly differentiated from the page numbers, either by distance or styling so that they can’t be mistaken for page numbers.

6. **Use First and Last links (where applicable)** – Provide the appropriate link or links. It’s not always necessary to add both of them or even any single one.

7. **Put First and Last links on the outside** – This is the common way of doing this and therefore it’s intuitive.
Rønn-Jensen [42] later revised the list where he suggested the following changes:

6. **First and Last links (where applicable)** – Provide a link back to the first page and don’t use a link to the last page, unless there *really* is a value for the end user on the last result page.

Rønn-Jensen [42] also suggested an additional point:

8. **Number of pagination links** – Provide a reasonable number of pagination links. Usually showing no more than 3 pages in each direction are enough.

**Filtering**

Lammi [44] has a few recommendations for data filtering implementations. The key points follow:

- **Select filters carefully** – Select a reasonable amount of filters which are the most useful to users and hide the others.
- **Order filters based on importance** – Important filters should come first. Important filter values may also come first, but sometimes alphabetical or numerical ordering is more appropriate.
- **Indicate what’s active** – Clearly indicate which filters are active and what they’re set to.
- **Display the number of occurrences in each facet item** – A common way of doing this is to show the number in brackets after the facet item.
- **Update results instantly** – In order to help users, update the search results instantly without reloading the page.
- **Visual feedback** – Tell the user what’s happening by for example showing a progress indicator when the results are being updated.

**Zebra Striping**

Zebra striping refers to the application of a faint shadow to alternate lines or rows in data tables or forms. Enders [45] conducted a study to determine if there are any reasons to use zebra striping.
Enders [45] found that zebra striping gave no significant improvement in accuracy and only marginal improvement in speed. She however found that some users used the mouse, a finger or something else to follow down columns and across rows in the absence of zebra striping. This is one of her points on why zebra striping still should be used [46]. She also adds that a big group of her test participants liked the zebra striping and another big group did not really care, so there is not much harm of adding zebra striping.

When using zebra striping, try to design the zebra stripes so that they will not cause an unnecessary amount of visual distraction. If not using alternating zebra stripes, a hover effect can be added to the rows of a table to increase the readability of the table. This type of visual feedback may however lead users to believe that the table rows are clickable. [21]

### 2.3.7 Information Dashboards

Information dashboards are useful to show information from one or more sources that need to be monitored regularly, or when an action might be needed based on a part of a large set of data.

Lammi [47] has compiled a list of guidelines

- **Distill and condense key data using summaries and exceptions** – Provide high-level summaries of the data. Tell what is happening, not why it is happening, but provide possibility for the users to drill down deeper into details. Focus on changes and exceptions in the most critical data. This is often the most important information to the users.

- **Fit the information on one page** – The purpose of a dashboard is to provide the key information from different sources quickly and easily to the user. Therefore the data should be shown on a single screen, preferably without scrolling.

- **Organize the content in groups** – Organizing the content into meaningful sections and subsections that reflect users’ needs makes scanning easier and helps to understand the content quickly.
• **Strive for simplicity** – Keep the dashboard clean and simple to make it easy to scan and understand. Use visual elements consistently and as little as possible, and highlight only data that needs attention.

• **Customize the dashboard for its users** – In order to be able to select the most important information it’s needed to understand the users and their goals. Personalization of the dashboard may be allowed.

• **Select the best mechanisms for clear and efficient data communication** – There are a lot of techniques for data communications. Use text, graphs, mind maps, icons, images, tables etc.

• **Test it with users** – Users will only use the dashboard if it feels useful and isn’t too difficult to use.

### 2.3.8 Usability Aspects of Ajax

The term Ajax is an acronym for “Asynchronous JavaScript and XML” [48] and describes a way to use JavaScript in a web page to communicate with a web server. This technique provides developers with the possibility of changing the content of web pages without fully reloading them.

Kluge et al. [27] studied the usability aspects of Ajax in web applications. They state that the application speed can be increased by for example only loading the part of a page that has changed rather than reloading the whole page. Ajax also makes a new level of smoothness and interactivity possible since the user interface can be available to the user while data is being loaded in the background. They also state that Ajax also makes new features possible. As examples they mention an input field with completion suggestion for the typed characters and progress indicators for data loading.

Ajax does however also come with some problems. First they mention that Ajax applications completely rely on JavaScript and hence are Ajax applications not accessible to users using browsers without JavaScript support. They also mention that strong efforts are necessary in order to reach a broad compatibility. Finally they state that since the URI does not change when Ajax content is loaded, the browsers’ Back and Forward buttons as well as bookmark function are rendered useless. [27]

Kluge et al. [27] mention that there however are workarounds to these problems. Providing users with application versions not using JavaScript makes the application accessible to more users. Current releases of Mozilla Firefox and Microsoft Internet Explorer support the implementation of custom functions for handling of the Back and Forward buttons. This problem, together with the bookmarking issue, can however be solved for more browsers by always coding the current page-state into the in-page link anchor marker “#”.

The study examined the completion times and satisfaction levels for two versions of two applications, one of each with Ajax functionality and one of each without it.
Overall they found that when wisely used, for example for auto-completion, Ajax can clearly improve the usability of web applications. [27]

**Keeping Users Informed**

Since Ajax by definition is asynchronous, content can be submitted or loaded while users are watching a page. In order to meet users’ expectations and conventions it’s therefore important to inform them of what is going on in the system. Jovanovic [49] lists a two ways to do this:

- **Simple loading indicators** – Loading indicators can be shown as simple text or as an image. Images are usually represented by a rotating animation.
- **Progress indicators** – If an operation needs a longer time to execute a progress indicator should be used. This provides real time information about the progress status.

For more information on when to use which indicator, see chapter 2.4.1 *Users and Response Times*.

Jovanovic [49] states that the indicator must be clearly visible. Depending on their purpose, indicators can be placed inside the same context as the element that started a request or they can be placed in a single position for all requests. He also says that particular elements or the whole user interface can be disabled in order to hinder users to use it during an Ajax request.

Jovanovic [49] finally says that when partially updating a page it is important to emphasize the outcome. The goal of this is to draw users’ attention to the updated area and enable them to easily confirm the update. This can be done by highlighting the updated area for a second, usually with a pale yellow background color.

### 2.4 Web Site Response Times

Response times are important for usability. Nielsen [2] states that keeping response times low is the most important design criterion for web pages.

There are of course many factors affecting the response times. Nielsen [2] lists the following factors:

- The throughput of the server.
- The server’s connection to the Internet.
- The Internet itself.
- The user’s connection to the Internet.
- The rendering speed of the user’s browser and computer.
All of these factors depend on hardware in some way but this thesis does not handle how to optimize hardware configuration. Instead techniques for how to reduce response times by optimizing front end loading performance are discussed. This does, according to Yahoo [50], often give the biggest performance increase with little effort.

2.4.1 Users and Response Times

Nielsen [2] has found three different reactions to response times, depending on the duration. These are described below:

- **0.1 seconds** is about the limit for having the user feel that the system is reacting instantaneously, meaning that no special feedback is necessary except to display the result.
- **1.0 seconds** is about the limit for the user’s flow of thought to remain uninterrupted, even though the users will notice the delay. Delays of this magnitude might make users lose the feeling of operating directly on the data. Getting a new page within a second means that the page appeared without undue delay.
- **10.0 seconds** is about the time limit for keeping the user’s attention. Getting a new page within 10 seconds, while annoying, at least means that the user can stay focused.

Nielsen [2] states that the satisfaction of users depends on their expectations as well as the actual response time performance. Therefore users should be helped to predict long load times.

Nielsen’s two statements above say that the longer the load time, the more important it is to indicate progress. If possible it’s best to show actual progress for load times, but an indicator showing that the page is loading is better than nothing. On the web, the latter is commonly represented by a rotating circle of dots. [49]

2.4.2 How to Lower Response Times

The Yahoo Exceptional Performance team has identified and compiled a list of best practices for making web pages fast [50]. These guidelines are not solely promoted by Yahoo but are widely used in the community. This list of guidelines will dominate this chapter.

**Make Fewer HTTP Requests**

According to Yahoo about 80% of the end-user response time is spent on the front-end, downloading external components in the page such as images, stylesheets (CSS), scripts and Flash. Reducing the number of components will reduce the number of
HTTP requests required to render the page. This is the most important guideline for improving performance for first time visitors and is the key to a better user experience. [50]

There are a few ways of minimizing the number of HTTP requests without modifying the pages’ design:

- **Combined files** – By combining all CSS into a single stylesheet and all scripts into a single script, the number of HTTP requests can be reduced. Combining files can be made a part of the release process. [50]

- **CSS Sprites** – Another way to reduce HTTP requests is to combine several images into a single image and use the CSS properties background-image and background-position to display the desired segment of the image [51]. This is the preferred method for reducing the number of image requests [50].

- **Inline images** – Image data can be embedded in web pages or stylesheets using the data: URL scheme. This increases the size of the documents, so the best practice is to use this in stylesheets which can be cached. Inline images are however not supported in all popular versions of all major browsers. [50]

- **Avoid redirects** – Using HTTP redirects always makes the browser send two HTTP requests, one request to the URL redirecting and another to the URL to which it was redirected, since redirects never are cached. The use of redirects should therefore be minimized. One of the most common wasteful redirects occurs when a trailing slash (/) is missing from a URL that should have one. Trailing slashes should therefore always be added to links where needed. [50]

Using these guidelines will decrease the number of HTTP requests required for any page with several images, stylesheets or scripts.

**Use a Content Delivery Network**

Again, since about 80% of the end-user response time is spent downloading the components of the page it’s a good place to optimize. A content delivery network brings content to users faster by serving content from servers distributed across multiple locations. [50]

**Cache Static Content**

Adding the HTTP header Expires with a date in the future to components, such as images, scripts or stylesheets, makes them cacheable. This does not reduce the load time the first time the components are loaded, but they will not be loaded again as long as they are still in the cache and the expiration date has not passed. [52] This will decrease the load time for subsequent page loads [50].
One way to make static components stay cached for a longer time is to implement “far future expire” policy by setting the Expires date to a far future date and renaming the file when it is modified. [50]

A way to further extend the use of the browser cache is to use the HTTP header ETag (entity tag). ETags were added to provide a mechanism for validating cached components that is more flexible than the last-modified date [52]. Browsers use their cached versions of the components if the previously received ETag matches the ETag of the same component on the server [50].

The caching techniques mentioned above should also be used for responses to Ajax requests where applicable [50].

Transfer Less Data

Another way to reduce response time is to reduce the amount of data sent over the Internet. There are a few ways to reduce the amount of data sent over the Internet without removing necessary information:

- **Gzip components** – Many modern browsers and web servers support compression of the transferred data. Compressing components using gzip often reduces the size of the data transferred. Files already compressed, such as most popular image formats and PDF files, should however not be compressed by the server. Doing this is a waste of computing power and might even increase file sizes. [50]

- **Minify JavaScript and CSS** – Minification is the practice of removing unnecessary characters from code to reduce its size [53]. The things usually removed in JavaScript and CSS are unneeded white space characters such as space, new line and tabs and in some cases comments [53][54]. Another way to reduce JavaScript size is to replace local symbols by shorter alternatives [55]. There are tools helping this process, which employ one or more of these practices [53][54][55].

- **Don’t scale images in HTML** – Don’t use a bigger image than needed just because width and height can be set in HTML. Instead, scale down the actual image. [50]

- **Optimize images** – There are a few techniques to optimize image file sizes. One is to convert GIFs to PNGs and see if there is a saving. According to Yahoo [50] more often than not, there is. There are also tools which losslessly can optimize sizes for specific file types [50], like pngcrush [56] for PNGs and jpegtran [57] for JPEGs.

- **Remove duplicate scripts** – One of the more obvious ways to reduce the data transferred is to remove duplicate scripts appearing several times. Removing the duplicates and only sending them once obviously reduces the amount of data transmitted. [50]
Make JavaScript and Stylesheets External
JavaScript and stylesheets common for several pages which the users generally visit should be external, instead of inlined in the page. By making them external they can be cached which will reduce load times on subsequent page loads. The only exceptions, where inlining is preferable, are home pages with few page views per session. This is because inlined JavaScript and CSS results in faster end-user response times with empty cache, since the number of HTTP requests in this case is reduced. [50]

Put Stylesheets at the Top
Putting stylesheets in the document’s HEAD tag makes pages appear to be loading faster. This is because putting stylesheets here allows the page to render progressively displaying whatever content it has as soon as it’s available. [50] This gives the user visual feedback of that the page is loading, which according to Nielsen is important [9].

Put Scripts at the Bottom
According to Yahoo [50] no parallel downloads occur while scripts are being downloaded. By putting scripts at the bottom of the document (but still inside the BODY tag) the browser can download all other resources in parallel [50]. Note that the HTTP/1.1 specification however suggests that browsers download no more than two components in parallel per hostname [58].

Reduce DNS Lookups
According to Yahoo it typically takes 20-120 milliseconds for DNS to lookup the IP address for a given hostname. DNS lookups are cached, but the use of too many hostnames will slow down the user experience. [50]

Using more than one hostname however has one upside. The number of parallel downloads can be increased by using more hostnames, because of the recommendation in HTTP/1.1 described in the previous section. Parallel downloads can decrease the time spent downloading but will also give more time spent on the DNS lookup. The Yahoo Exceptional Performance team recommends the use of no more than four hostnames. [50]

Avoid CSS Expressions
CSS expressions are a way to dynamically set CSS properties in Internet Explorer 5 and later. These expressions give great possibilities but are evaluated very frequent. They are for example evaluated when the page is rendered and resized but also when the page is scrolled and even when the mouse is moved over the page. Moving
the mouse around the page can easily generate more than 10,000 evaluations, which can make the page slow. [50]
2.5 Prototype Evaluation Method

The prototype will be created based on the usability guidelines above, and will then be evaluated from a usability perspective. The purpose of this evaluation is to find the credibility of the guidelines. This will be done by evaluating the prototype using a recognized evaluation method. If a violation is found, and if that violation is based on a guideline, the guideline is considered to be invalid.

Heuristic Evaluation – a systematic inspection of a user interface design for usability [9] – is the selected evaluation method.

2.5.1 Heuristic Evaluation

Heuristic evaluation involves having a small set of evaluators examine the interface and judge its compliance with recognized usability principles (the “heuristics”) [9]. These heuristics should be general rules describing common properties of a usable interface.

Heuristic evaluation is a very efficient method for finding usability problems, meaning that many usability problems can be found with little effort [59]. By using only a few evaluators and a quite limited time, many usability problems can be found. Enea [5] will provide the evaluators and because of this requested that an efficient method were used. This is the main reason why heuristic evaluation was chosen.

Nielsen [9] writes that the evaluation is performed by having each individual evaluator inspect the interface alone. The evaluator goes through the interface several times and inspects the various dialogue elements and compares them with a set of recognized usability principles. In addition to this checklist, the evaluator is also allowed to consider any additional usability principles or results that may be relevant for any specific dialogue element. Only after all evaluations have been completed are the evaluators allowed to communicate and have their findings aggregated. This procedure is important in order to ensure independent and unbiased evaluations from each evaluator. The results can be recorded as written reports from each evaluator.

Nielsen [9] also states that even though it’s preferable to use usability specialists (with domain knowledge) as evaluators, it is possible to conduct a successful heuristic evaluation performed by people with little or no usability expertise. More experienced evaluators however usually tend to find more of the existing usability problems. According to a study by Nielsen [6] the proportion of usability problems found increases with the addition of more evaluators, but the added gain for each new evaluator decreases as new evaluators are added. Hence adding more evaluators will give a better result, but adding too many is not cost-effective. Nielsen [6] recommends the use of at least three evaluators, but since this evaluation is done without usability experts, six evaluators are used.
Nielsen calls the heuristic evaluation a discount usability engineering method, which means that it’s not guaranteed to provide “perfect” results or to find every last usability problem in an interface [6]. This is the downside of the method, traded for the efficiency.

### 2.5.2 Usability Heuristics

The ten usability heuristics developed Nielsen and Molich [60] as described [9] and summarized [10] by Nielsen are listed below:

1. **Simple and Natural Dialogue** – User interfaces should be simplified as much as possible and should match the users’ task in as natural a way as possible, minimizing navigation. The optimal is to present exactly the information the user needs and no more at exactly the time and place where it is needed.

2. **Speak the Users’ Language** – Words, phrases and concepts should not be system-oriented but familiar to the user. Real-world conventions should be followed and information should appear in a natural and logical order.

3. **Minimize User Memory Load** – Make objects, actions and options visible to minimize the users’ memory load. Users should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

4. **Consistency** – Users should not have to wonder whether different words, situations or actions mean the same thing. Follow platform conventions.

5. **Feedback** – The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

6. **Clearly Marked Exits** – Users does not like to feel trapped by the computer. In order to increase the users feeling of being in control of the dialogue, the system should offer the user an easy way out of as many situations as possible.

7. **Shortcuts** – Having buttons or other shortcuts available to access important or frequently used functions where they are needed makes frequent users operate the system faster.

8. **Good Error Messages** – Error messages should be expressed in plain language (no codes), precisely indicate the problem and constructively suggest a solution.

9. **Prevent Errors** – Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

10. **Help and Documentation** – Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the users’ tasks, list concrete steps to be carried out and not be too large.
3 Creating the Prototype

3.1 Requirements

The purpose of the prototype, from this thesis’ point of view, is to implement many of the guidelines compiled in the thesis and thus allowing the validity of the guidelines to be tested.

The purpose of the prototype, from Enea’s point of view, is to be a good example of a usable web application demonstrating possible use cases of their M2M platform. The particular case selected for this thesis work is a snowplow fleet monitoring system with a few key issues to solve. According to specifications [61], the prototype should:

- Show performed routes on a map with the ability to show the following:
  - The vehicles’ velocities on the different parts of the recorded routes
  - The time since the services were performed
- Allow filtering of data on the map based on the following parameters:
  - Services performed: plowing, salting or sanding
  - Time since the service was performed
  - The movement speed
- Show status of the vehicles in a list with the following information:
  - The state of the vehicle: running or stopped
  - The state of the car alarm: triggered, active or passive
  - More detailed information such as plowed distance and salt tank level
- Allow filtering of data in the list based on the following parameters:
  - The state of the vehicle: running or stopped
  - The state of the car alarm: triggered, active or passive

The prototype should be implemented in Python using the Django web framework [62].

3.2 Design and Implementation

The prototype is not designed or implemented according to any specific process. It is however designed based on as many as possible of the applicable guidelines listed in Appendix A: Guidelines.
Design decisions connected to the guidelines will be explained below, where the guidelines are referenced by their number.

### 3.2.1 Common Layout

The different pages have a consistent layout (1.1) with a logo in the upper left corner (5.1) which links back to the first page in its initial state (5.2). The pages are divided into clear areas (1.3, 1.4) and do not have unnecessary use of color or animations (1.7).

The navigation interface is persistent and always shown on the top of the page (5.3, 5.6) and combines blue links with a tab showing where the user is located (5.4, 5.7). Links are vivid blue and underlined with descriptive titles (6.1, 6.2 & 6.3). No other text is colored nor underlined (6.4, 6.5) except the logotype and highlighted important information (8.5). All links are functional and hence have the same color, independent of whether they are visited or unvisited (6.6).

![Logo and navigation](image)

**Figure 11. Logo and navigation.**

### 3.2.2 The Map View

The map view of the prototype consists of filters and settings at the top, a legend for the map to the left and a map taking up the rest of the window. The page design is fluid and follows the window when resized, eliminating the need of scrolling (1.6, 9.2).
The filters at the top affect the lines drawn on the map. The filters reside in a box with a slightly darker background than the rest of the page, making the filter box more prominent (1.2) as well as making the filters look related (1.3). Standard controls are used where possible (7.1, 7.2 & 7.3), with the addition of the slider control which was added to simplify the interaction with a mouse (7.4, 8.17). The standard controls always indicate the value of the active filter (8.16).

The filters are by default set to show all data (7.15), and when a filter value is changed, the map is automatically updated (8.19). An indicator tells the user that the system is loading while the map is being updated (10.3).
3.2.3 The List View

The list view of course contains a list but is in some other ways similar to the map view above. For example, standard controls are used for filtering and the filter box is styled the same way (1.1). Some of the guidelines applied to the map view have thus also been applied to the list view.

![Screenshot of the prototype's list view.](image)

The list by default only shows what’s important to the users\(^2\) but gives users the ability to see more details (8.1). The list is light on visual elements, with the highlighted important data being the only exception (8.5). No pagination is used because it would not add any value (8.10).

---

\(^2\) According to the specifications in chapter 3.1 Requirements.
4 Prototype Evaluation

4.1 Procedure

The evaluators, selected by Enea [63], will conduct the evaluation on their own. They will be sent an email of which a copy can be found in Appendix B: Evaluation Instructions. The email contains a description of what the evaluator is asked to do as well as the usability heuristics in chapter 2.5.2, formulated as statements.

They will be asked to state if they agree to the statements. If they do not agree, a description of what violates the principle as well as a motivation is requested. If they do agree, the principle is considered to be followed. The fewer principles violated, the more usable is the prototype considered to be.

4.2 Statements

The evaluators will be presented to the following list of statements:

1. Simple and Natural Dialogue
   a. The user interface is as simplified as possible.
   b. Navigation is minimized.
   c. The users’ tasks are matched in a natural way.
2. Speak the Users’ Language
   a. Words, phrases and concepts are familiar to the user rather than system-oriented.
   b. Information appears in a natural and logical order.
3. Minimize User Memory Load
   a. Objects, actions and options available to the user are visible.
   b. Users do not have to remember information from one part of the dialogue to another.
   c. Instructions for use of the system are visible or easily retrievable where appropriate.
4. Consistency
   a. Conventions are followed.
   b. Users do not have to wonder whether different words, situations or actions mean the same thing.
5. Feedback
a. The system always keep users inform about what is going on, within reasonable time.

6. Clearly Marked Exits
   a. The system offer users an easy way out of as many situations as possible.

7. Shortcuts
   a. Buttons or other shortcuts are giving access to important or frequently used functions are available where needed.

8. Good Error Messages
   a. Error messages are expressed in plain language.
   b. Error messages precisely indicate the problem and constructively suggest a solution.

9. Prevent Errors
   a. Error-prone conditions are eliminated, or users must confirm before committing error-prone conditions.

10. Help and Documentation
    a. Help and documentation is available where necessary.

### 4.3 Answer Summary

This chapter contains a summary of the gathered usability flaws from the evaluation. The complete list of answers is included in *Appendix C: Evaluation Answers*.

Two problems in the list below are highlighted with bold text. These two are the only usability flaws related to any of the guidelines.

<table>
<thead>
<tr>
<th>Number</th>
<th>Identified # times</th>
<th>Usability problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>The relation between “Services performed” checkboxes is logical AND, but logical OR would be more obvious</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>There is no easy way to get back to map data when the map has been navigated somewhere else</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>The “Time since” option for route coloring needs an explanation</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>Opened plow details in the list view are forgotten when filters are updated or when the page is reloaded</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>The legend items in the map view looks clickable</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>The different alarm states (Triggered/Active/Passive) are unclear and need explanations</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>Users may want to see the current location of the plows</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Inconsequent use of Active/Passive and Enabled/Disabled for alarm states in the list view</td>
</tr>
<tr>
<td>No.</td>
<td>Count</td>
<td>Issue Description</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>-------------------</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>The “Currently active” filter is unclear and needs explanation</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td><strong>The link target of the logotype is not what was expected</strong></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>The &quot;Services performed&quot; setting in the map view needs an explanation</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>The &quot;Time since&quot; text boxes does not validate the data entered</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>Alarm state is not visible in map legend, but it probably should</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>The “File not found” error messages are strange and not helpful</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>Filters are reset when pages are reloaded</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>There is no way to tell which end is the head or the tail of a route</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>Instructions are missing in the whole application</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>“List view” is not a descriptive name</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>The list view is not organized as a table, but as a list</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>The list view does not show enough information</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>The language in the map is Swedish, while the rest of the application is in English</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>There is no obvious way to see the relation between the plows in the map view and in the list view</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>Slider filters should instantly update the map view when changed, not when released</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>Vehicle is a better word than car in car alarm</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>There is no description of what the routes show</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>There should be a symbol on the sliders to show that they are “draggable”</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>Slider handles may obstruct each other</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>Slider handles should be able to be pushed by dragging the other handle</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>Slider handles should be able to be pushed by entering a lower/higher number in the text boxes</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>The “Time since” slider should not be reversed</td>
</tr>
</tbody>
</table>
Usability problem number 5 is a direct and obvious violation of guideline 6.8; “Nothing that is not clickable looks clickable”.

Usability problem number 10 is however a problem contradicting a guideline 5.2; “The site logo links back to the home page or the first page the users see when they’re logged in”. Although this design is based on a guideline, the violation may be a design or implementation issue rather than an invalid guideline. The reason of this confusion may be that the logo is the Enea company logo and not a logo specifically designed for the prototype. Another reason might be that the map view, to which the logo links, might not be the obvious start page of the application. While it is hard to completely disregard this suggested violation, Nielsen’s research backing up the guideline is more substantial than this evaluation.

The other identified usability problems do not have an obvious connection to any guideline, so while they may be valid, they have no impact on the validity of the list of guidelines in its current form.
5  Conclusions

While the evaluators found 32 different usability issues, only one of the found violations was directly related to a guideline. This leads to two conclusions. Firstly, while the list of guidelines may be incomplete, no guidelines were found to be invalid (with one possible exception, discussed below). Secondly, the amount of usability issues found by the evaluators makes it obvious that it is not sufficient to create a web application only based on these guidelines, even if it’s a small application. Some amount of usability testing is needed to ensure that the web application really is usable. This is actually supported by Nielsen, who says that general guidelines never can fully replace usability testing, since there always are domain specific issues [4].

The only violation directly related to a guideline was found by Error! Reference source not found., referring to the link target of the logo. While it is hard to completely disregard this possible violation, Nielsen’s extensive research backs up the guideline which leads me to believe that the issue is a design or implementation issue rather than an invalid guideline. This is more extensively motivated under Error! Reference source not found. in chapter Error! Reference source not found. Error! Reference source not found.. The guideline will thus not be removed from the final list.

The full list of guidelines can be found in Appendix A: Guidelines.

5.1  Method Discussion

The list of guidelines is not proven to be valid, in a strict sense, but rather not found to be invalid. The guidelines are also not proven to give a better web application in this thesis. However, the thesis is based on articles proving the benefits of most of the guidelines by themselves, and the rest of the guidelines are accepted as conventions.

Some articles, especially on the Internet, may contain personal opinions rather than scientific, peer reviewed facts but efforts have been made in order to validate all information gathered.

Since the list itself was not tested, but a prototype based on the list, extra uncertainty is added. This is however probably the best way to evaluate the guidelines. To further examine the credibility of the list of guidelines, more web applications based on the guidelines can be evaluated.
It is possible that if a more extensive (and expensive) evaluation method would have been used, more usability problems would have been found. Giving the evaluators more domain knowledge may possibly also have given another result. Having a set of evaluators with another level of computer familiarity might also have affected the results. The computer familiarity level does however match the level for the average user of this prototype.

A lot of the information is based on Nielsen’s work, and the evaluation method together with the heuristics is also partly created by Nielsen. One could argue that it is a problem that Nielsen’s recommendations would always prove to be valid when tested with his own evaluation. I however don’t think that this makes his recommendations less valid, and doing a wide web usability study like this one without referring to Nielsen’s work would be challenging.

### 5.2 Further Research

There are still areas related to the ones examined which deserve further attention. The list of guidelines may for example be extended with recommendations for error messages, both for form validation and general errors. Apart from extending the list, it may also be interesting to further test the validity of the list of guidelines in its current form. This could be done by for example testing a more complex application built based on the guidelines.

Some of the evaluators complained about settings not being saved between page views. The persistence of user choices in web applications may hence be an interesting subject to research. What user actions should be remembered, and when?

Another area of investigation which is becoming more and more interesting is web applications in mobile devices, with for example small screens or touch screen interfaces. What are the requirements on web applications designed for such devices?

It could also be interesting to examine how web application performance is best increased, after the frontend performance techniques described in this thesis have been applied.
Bibliography


http://www.contentwithstyle.co.uk/content/fixing-the-back-button-and-enabling-bookmarking-for-ajax-apps.


APPENDIX A: Guidelines

Below follows the list of guidelines compiled in the thesis work. The guidelines can be used as a checklist to improve the usability of web applications. All guidelines are not applicable to every web application case, but many are general and apply to most cases, and even regular web sites.

This list is one part of what Enea wanted from the thesis work.

1 Page Layout
These guidelines describe general page design and are based on chapter 2.3.2 Page Layout and Content.

1.1. Pages have a consistent design
1.2. Things that are more important are more prominent
1.3. Things that are related look related
1.4. Things that are part of something are visually nested
1.5. Conventions are used where possible
1.6. Pages have a vertical layout which does not require horizontal scrolling
1.7. Colorful or animated areas are avoided because of users “banner blindness”

2 Writing for the Web
These guidelines are for text and are based on the section Writing for the Web of chapter 2.3.2 Page Layout and Content.

2.1. Texts are short with no needless words
2.2. Important keywords are highlighted
2.3. Pages have meaningful sub-headings
2.4. Bulleted lists are used where applicable
2.5. Paragraphs only contain one idea
2.6. Pages with a lot of text start with the conclusion, according to the inverted pyramid style

3 Home Page
These guidelines are based on the section The Home Page of Web Sites of chapter 2.3.2 Page Layout and Content. Some of the guidelines assume that the home page is the page convincing new users to use the application.

3.1. The home page shows the identity and mission of the site.
3.2. The home page gives an overview of what the site has to offer, both in terms of content and features.
3.3. The home page promotes the newest, best or most popular content or features.
3.4. If the site is large the home page contains a prominently displayed search box.
3.5. The home page contains parts that sometimes are updated.
3.6. The home page contains short-cuts to the most frequently requested content.
3.7. If the site uses registration, the home page allows users to register or sign in.
3.8. The home page shows users where to start.
3.9. The home page establishes credibility and trust.
3.10. If the site does not have to warn users about its content, the site contains no splash screens.

4  Credibility

These guidelines state how web applications are given credibility, and are based on the section Credibility of chapter 2.3.2 Page Layout and Content.

4.1. The accuracy of the information is easily verified – citations, references or source material is provided.
4.2. The site contains the physical address of the organization
4.3. Expertise in the organization is highlighted
4.4. It’s shown that there are honest and trustworthy people behind the site
4.5. The contact information is clear
4.6. The site has a professional or appropriate design
4.7. The site is easy to use and useful
4.8. The sites content is often updated
4.9. The site contains restrictive use of advertisements and no pop-up ads
4.10. The site has no errors, such as spelling errors or broken links.

5  Navigation and Identity

These guidelines give recommendations on how to make the navigation of web application usable. They are based on chapter 2.3.3 Navigation.

5.1. Every page contains the sites logo, which is best placed in the upper-left corner
5.2. The site logo links back to the home page or the first page the users see when they’re logged in
5.3. The navigation shows the site hierarchy so that users understand what the site contains
5.4. The navigation always show users where they are
5.5. If the site is large and hierarchically structured, breadcrumbs are used
5.6. Navigation is placed on the same place and work the same way on all pages, possibly except the home page and pages containing nothing but forms.
5.7. Navigation is not drastically different from the navigation on other web sites
5.8. The web site does not break the browsers back button by unnecessarily opening new windows, using HTML redirects or Ajax implementations without browser history support.
5.9. The domain name of the site is easy to remember and easy to spell
5.10. The URLs of the site are persistent, short, easy to type and visualize the site structure

6 Clickable Objects
These guidelines describe clickable objects, such as links, buttons and other navigational elements. They are based on chapter 2.3.4 Clickable Objects.

6.1. Links and buttons are styled so that they catch the user’s attention
6.2. Links and buttons have descriptive texts, and not for example “click here”
6.3. Links begin with the most important keyword
6.4. Underlining is reserved for links
6.5. Colored text, especially blue, is avoided unless it’s a link
6.6. Visited links have a duller and less vivid color than unvisited hyperlinks, for links which refer to content rather than functionality.
6.7. Everything that is clickable looks clickable
6.8. Nothing that is not clickable looks clickable
6.9. Clickable areas are appropriately sized
6.10. Double-clicking will not cause problems
6.11. Primary actions are more prominent than secondary actions
6.12. Primary actions are positioned so that they follow users’ natural flow

7 Forms
These guidelines describe forms, form elements and labels. They are based on chapter 2.3.5 Forms.

7.1. The site uses the right form control types for the tasks
7.2. Radio buttons are used when selecting from a list of mutually exclusive items, if the list is reasonably small
7.3. Checkboxes are used when selecting from a list of non-mutually exclusive items
7.4. Text input fields have a size appropriate for the data asked for
7.5. Reset buttons are only used if really needed (which they rarely are)
7.6. Form labels are placed close to their associated controls
7.7. Form labels are programmatically associated with their controls
7.8. Labels of checkboxes and radio buttons are placed to the right of the controls
7.9. The best position for form labels are in most other cases directly above their controls
Appendix A: Guidelines

7.10. Required fields are marked, preferably by placing an asterisk or the word “required” near their labels
7.11. The system allows users to enter data in a flexible way
7.12. Forms shows hints on how data should be entered if there are requirements
7.13. Forms detects errors on the server-side and optionally also on the client-side
7.14. Users do not have to enter the same data more than once
7.15. Forms gives sane default values where possible
7.16. The first form control on pages is automatically focused
7.17. Long forms have a vertical layout rather than horizontal
7.18. When forms are split up on multiple pages, progress and what’s left is shown
7.19. Site navigation may be hidden on pages where filling out a form is the main action

8 Tables and Lists of Information
These guidelines describe tables and other lists of information, along with filtering and pagination. The guidelines are based on chapter 2.3.6 Tables and Lists of Information.

8.1. Tables only show what users really have to see, but gives the ability to see details
8.2. Numeric data have an appropriate level of precision
8.3. Data is summarized when possible
8.4. Visual noise is reduced – all interface elements support structure, organization and readability.
8.5. Important data is highlighted
8.6. Proper alignments are used
8.7. For long tables, table headers float so that they are always visible
8.8. Tables have a logical sort order by default
8.9. Tables allow custom sorting where applicable
8.10. Long tables can be split up using pagination or endless scrolling if it adds any value to the users
8.11. Pagination links have large enough clickable areas and have space between them
8.12. Pagination has links to the first, previous and next page. A link to the last page is only added if really needed.
8.13. The number of pagination links is reasonable. Usually 3 pages in each direction are enough.
8.14. Tables have filtering options where applicable
8.15. Filters are ordered based on importance, with the most important first. Less useful filters are hidden.
8.16. It’s clearly indicated which filters are active and what they are set to
8.17. Filter controls are intuitive, and can preferably be manipulated using the mouse
8.18. The number of occurrences in each facet item is shown
8.19. Filtered results are updated instantly without reloading the page
8.20. Zebra striping may be used but will probably not affect the usability
8.21. If zebra striping is used, no unnecessary visual distraction is caused by using alternating styles that are too different

9 Information Dashboards
These guidelines apply to information dashboards, which gives users the ability to monitor or make decisions based on data. The guidelines are based on chapter 2.3.7 Information Dashboards.

9.1. High-level summaries and exceptions in key data is provided
9.2. The information is fit on one page, preferably without scrolling
9.3. Content is organized in groups
9.4. The dashboard is clean and simple
9.5. Consider allowing users to customize the dashboard
9.6. Good mechanisms for clear and efficient data communication are used. Examples are text, graphs, mind maps, icons, images and tables.

10 Response times
These guidelines describe how response times should be handled as well as how load times can be reduced. They are based on chapter 2.4.2 How to Lower Response Times.

10.1. The system helps users predict long response times
10.2. The system shows progress for long response times, if possible
10.3. The state of the system during Ajax requests is shown
10.4. The number of HTTP requests required to use the site is reduced by:
   • combining stylesheets into a single file
   • combining JavaScripts into a single file
   • using CSS sprites
   • avoiding redirects
10.5. A content delivery network is used for static content
10.6. Static content is cached
10.7. The amount of data transferred when using the site is reduced by:
   • gziping components
   • minifying JavaScript and CSS
   • optimizing images
   • don’t scaling down images in HTML but using a smaller image
   • removing duplicate scripts
10.8. Scripts and stylesheets are external so that they can be cached
10.9. Stylesheets are put at the top of the page code
10.10. Scripts are put at the bottom of the page code
10.11. No more than four hostnames are used to reduce the amount of DNS lookups required
10.12. CSS expressions are avoided
APPENDIX B: Evaluation Instructions

Hello!

Thank you for participating in the M2M snowplow prototype evaluation. The prototype demonstrates an implementation of a snow plow fleet monitoring system, allowing users to view and filter the fleet’s routes and information on the plows.

This evaluation method is called Heuristic Evaluation, and it’s done in such a way that you check the compliance of every element (of the site’s pages) with the list of statements below. The best result is achieved if you check every element more than once. If you find any violated statements, please write down the following:

- The element or elements that violated the statement
- The statement that was violated
- A motivation, why you think it was violated

Note that if you find violated usability principles which are not in the list below, you may include them in your answer. Please do your best to identify usability problems.

The prototype should be viewed with a recent version of Firefox and can be found here: (URL removed)

The usability statements are the following:

1. Simple and Natural Dialogue
   a. The user interface is as simplified as possible.
   b. Navigation is minimized.
   c. The users’ tasks are matched in a natural way.

2. Speak the Users’ Language
   a. Words, phrases and concepts are familiar to the user rather than system-oriented.
   b. Information appears in a natural and logical order.

3. Minimize User Memory Load
   a. Objects, actions and options available to the user are visible.
   b. Users do not have to remember information from one part of the dialogue to another.
   c. Instructions for use of the system are visible or easily retrievable where appropriate.

4. Consistency
   a. Conventions are followed.
   b. Users do not have to wonder whether different words, situations or actions mean the same thing.

5. Feedback
a. The system always keep users inform about what is going on, within reasonable time.

6. Clearly Marked Exits
   a. The system offer users an easy way out of as many situations as possible.

7. Shortcuts
   a. Buttons or other shortcuts are giving access to important or frequently used functions are available where needed.

8. Good Error Messages
   a. Error messages are expressed in plain language.
   b. Error messages precisely indicate the problem and constructively suggest a solution.

9. Prevent Errors
   a. Error-prone conditions are eliminated, or users must confirm before committing error-prone conditions.

10. Help and Documentation
    a. Help and documentation is available where necessary.

If you run into any problems please contact me. Otherwise, please mail your results back to me when you are done. Answers may be recorded in Swedish or English, but may be translated to English when presented in the thesis.

Thank you for your participation!

/David
APPENDIX C: Evaluation Answers

The answers from the evaluators follow below. Some of them were translated from Swedish to English for the sake of the report. The evaluators’ findings will be presented in the following form:

Violation number

Element: The element or elements violating a statement
Violation: The statement or statements violated
Reason: The evaluator’s motivation to why he or she thinks the element(s) violates the statement(s).

After some findings there is a comment discussing the finding.

For the sake of anonymity, the evaluators will not be called by their real names.

5.2.1 Evaluator A

Violation 1

Element: Slider filters, map view
Violation: 1. Simple and Natural Dialogue
Reason: The sliding bars handles have no marking. If you are not very familiar with this kind of user interface, it might not be obvious that they could be moved. Please fit them with arrows \([<->]\) or similar.

Violation 2

Element: Slider filters, map view
Violation: 1. Simple and Natural Dialogue
Reason: If both sliders are set to the same value, one of them is obstructing the other to move. The “upper” slider can be moved in one way, but the “lower” slider cannot be moved at all.
Violation 3
Element: Slider filters, map view
Violation: 1. Simple and Natural Dialogue
Reason: I would like to be able to take one slider and push the other if a greater value is wanted.

Violation 4
Element: Slider filters, map view
Violation: 1. Simple and Natural Dialogue
Reason: The sliders have a lower and an upper value. If you, by typing a number in the textbox for the lower value, enter a value larger than the upper value, the lower value only get raised to the upper, not to the value set by the user. Both should move.

Violation 5
Element: Time since filter, map view
Violation: 1. Simple and Natural Dialogue
Reason: It is illogic to have a reversed direction on the “Time since” scale. At least, that’s what I thought at first glimpse.

Violation 6
Element: Legend, map view
Violation: 1. Simple and Natural Dialogue
Reason: In the “Legend” part of the screen I got the feeling I was able to select one of the cars, but that didn’t work. Neither could I click on the “Running” label to see running status. I would like to get to a properties view of that car.

Violation 7
Element: Filter labels, map view
Violation: 2. Speaks the User’s Language
Reason: In the “Color routes by” radio button selectors the choice “Time since” sort of hangs in the air. Time since what? It could have been “Time since last activity”.
Violation 8
Element: Time since filter, map view
Violation: 4. Consistency
Reason: The slider for the “Time since” value cannot be set by entering a value in the text box... Ah, wait. You can, but the slider values are positive in the wrong direction. By entering a larger value you get further to the left. Weird.

This is by design, but if it should be considered a usability flaw, it is the same flaw as described in Error! Reference source not found..

Violation 9
Element: Car alarm filter, list view
Violation: 4. Consistency
Reason: The “Car alarm” selection radio buttons are called “Active” and “Passive” while the corresponding property in the detail view of the Car is “Enabled” and “Disabled”.

Violation 10
Element: Alarm, list view
Violation: 4. Consistency
Reason: The “Triggered” selection choice corresponds to a label of “Alarm” on the Car.

Violation 11
Element: List view
Violation: 4. Consistency
Reason: When you open all detail views and then select a different option, all detail views are closed again. I would like them to stay open.

Violation 12
Element: –
Violation: 5. Feedback
Reason: The system is very fast. I found no use of, nor need of (right now), any feedback for lengthy operations. That would be necessary if they existed though.
Appendix C: Evaluation Answers

This will not be considered as a usability flaw.

Violation 13

Element: –
Violation: 6. Clearly marked exits
Reason: I found no Exits at all, but then again, I found no Entrances either.

This will not be considered as a usability flaw.

Violation 14

Element: –
Violation: 7. Shortcuts
Reason: I found no shortcuts. It would be nice to be able to switch view by a keyboard shortcut.

Violation 15

Element: –
Violation: 10. Help and Documentation
Reason: There is no Help and no Documentation at all. In fact, there are not even brief instructions at hand. There could have been context driven pop-ups when you hover over controls (as in the Google Map frame).

5.2.2 Evaluator B

Violation 1

Element: Map view
Violation: 2. Speak the users’ language
Reason: Mixed language, Swedish & English. (Swedish in the map, the rest is in English)
Violation 2

Element: –
Violation: 3. Minimize user memory load
Reason: Most of the things are self explanatory, which is good. Furthermore there are virtually no things depending on each other which need to be remembered. There are no instructions at all but the question is if they are needed, and I don’t need any instructions.

This is not considered to be a violation of the principle.

Violation 3

Element: –
Violation: 4a. Conventions are followed
Reason: I think conventions are followed in a good way.

This is not considered to be a violation of the principle.

Violation 4

Element: List view
Violation: 4b. Users do not have to wonder whether different words, situations or actions mean the same thing
Reason: The list view can be filtered on “Currently active” or “Car alarm”. These two categories are on somewhat different levels. What is currently active? The alarm, is the snowplow running, or what? Car alarm is more obvious. It is however obvious, when looking again, that the purpose is whether the snowplow is active or not.

Violation 5

Element: –
Violation: 5. Feedback
Reason: User feedback is given but if you are on a slow connection update is shown instantly. That is good.

It is unclear whether the evaluator meant that this is a violation or not.
Violation 6
Element: Map view
Violation: 6. Clearly marked exits
Reason: I would not say there is any exit, but it is possible to get out of most of the situations. What I would possibly want is some kind of “get back to plow data” functionality in the map. It is there possible to go far away and then not be able to find the plow data again.

Violation 7
Element: File not found messages
Violation: 8a. Error messages are expressed in plain language
Reason: If you enter an invalid URL you get a strange Django error message. Imagine www.snowplow.com and www.snowplow.com/list, but if you write something like www.snowplow.com/bad_url it would be better if users were shown a better error message. This is the only error message I can find.

Violation 8
Element: Map view, “time since” filter
Violation: 8b. Error messages precisely indicate the problem and constructively suggest a solution
Reason: If you enter 26,3 in a “Time since” text box, the page hangs while updating and no error message is shown. The same thing happens when other erroneous values are entered, such as characters or symbols.

Violation 9
Element: Map view, “time since” filter
Violation: 9. Prevent errors
Reason: Maybe users should only be allowed to enter valid data in the “Time since” text boxes, or get a message stating that something is wrong if erroneous data is entered.

Violation 10
Element: General
Violation: 10. Help and documentation
Reason: I could not find any help at all, but is it needed? Maybe an explanation should be provided regarding for example “time since”.
5.2.3 Evaluator C

Violation 1
Element: Radio Button “Time Since”
Violation: 2. Speak the users’ language
Reason: The “Time since” sentence does not say what it refers to. Time since what?

Violation 2
Element: Map view – routes
Violation: 10. Help and documentation
Reason: It is not clear what the routes are showing. Is it the routes that has been driven by the vehicles or the routes that they are planned to drive?

While this obviously was unclear to the evaluator, an operator working in the system will most likely know what the purpose of the system is. While this is a usability flaw, it is also a sign of the lack of information about the purposes of the system to the evaluators.

Violation 3
Element: Map view
Violation: 5. Feedback
Reason: It is not possible to tell what is the head or tail of a route plotted on the map. There are no indications of where the vehicles are right now.

Violation 4
Element: Service Performed check-boxes
Violation: 1. Simple and natural dialogue
Reason: The natural way to see the check boxes is that they filter out certain operations that have been performed by the plows. If I would like to know where has been sanded or salted I would naturally click the Salting and the Sanding boxes. There is nothing in the GUI that tells if this is an “AND” or an “OR” operation between the checked services.
Violation 5
Element: Map view
Violation: 3. Minimize user memory load
Reason: If you zoom out and move away from the area where there is plotted data, it is hard to find your way back in a simple way. There should be a way of focusing and zooming the map to the data that is available for a given vehicle. This problem will be more apparent when you have plows driving separated over a larger area. There should be a way of clicking on something in the legend and have the map focus on it.

Violation 6
Element: “Currently active” text string in List view
Violation: 2. Speak the users’ language
Reason: The phrasing is not clear. How to interpret “Currently active stopped” – is it active or stopped? “Current activity” or something similar might be a better phrasing.

Violation 7
Element: “Car alarm” text string in List view
Violation: 2. Speak the users’ language
4. Consistency
Reason: Are we talking about cars or plows? A plow is most often a tractor, not a car. Vehicle is a better wording.

Violation 8
Element: “Car alarm” radio buttons
Violation: 10. Help and documentation
Reason: What is the difference between a triggered alarm and an active alarm? What is a passive alarm?

Violation 9
Element: Plow list in List view
Violation: 3. Minimize user memory load
Reason: There is nothing that tells which plow is which. In the details of each plow there should be a map that shows where it is right now on a map in order to identify what plow it is.
Violation 10

Element: Plow list in List view
Violation: 1. Simple and natural dialogue
Reason: The details of the plows in the list are hidden every time a new filter is selected at the top of the page. If I have clicked to show all details, I would like it to remember this so I don’t have to click the “show all” button every time I make a new filter selection.

Violation 11

Element: Plow driver
Violation: Traffic violation
Reason: You are only allowed to drive at maximum 50 km/h on Wallenbergs Gata. :) This refers to the fact that some snowplows had a monitored velocity of 70 km/h on a street where the speed limit is 50 km/h. This violation is hence not regarded as a usability issue, and the smiley emphasizes that the addition is of the more humorous nature.

5.2.4 Evaluator D

Evaluator D commented that it was difficult to perform the evaluation since she does not know much about snowplows and is no usability expert. She also said that since the answers only highlight the flaws of the system, the evaluation results seem quite negative but she actually thinks that the system is very good looking and relatively easy to use. She however misses instructions for how the system is supposed to be used and what the purpose of the system is.

Violation 1

Element: Map view
Violation: 1a. The user interface is as simplified as possible
Reason: I don’t see any reason to show velocities in an application like this one, so to me that option seems unnecessary, but on the other hand I don’t really know what the application is for. Maybe the velocity is really important to know, if for example a snowplow ran off the road because it was going too fast. Conclusion: I can’t decide this since I don’t know the full purpose of the system.

Displaying velocities might in many cases make no sense, but since this was part of the prototype requirement specification this will not be considered as a usability flaw.
Appendix C: Evaluation Answers

Violation 2
Element: –
Violation: 1c. The users’ tasks are matched in a natural way.
Reason: I cannot say anything about this since I don’t really understand what the users’ tasks are.

This will not be considered as a usability flaw.

Violation 3
Element: Car alarms
Violation: 2a. Words, phrases and concepts are familiar to the user rather than system-oriented
Reason: It’s difficult for me to tell since I’m not familiar with the terms used by someone who monitors snowplows. For example I don’t understand what the different alarm states mean. If an alarm is trigged, what does that mean?

Violation 4
Element: The legend in the map view
Violation: 3a. Objects, actions and options available to the user are visible
Reason: It seems like an important part of the system is to make sure users know what alarms have been triggered. Because of this, I think triggered alarms should be shown in the map view, which is the first view shown to the users. The best place to show them is probably in the “legend”, so I guess that element violates this principle.

Violation 5
Element: List view
Violation: 3b. Users do not have to remember information from one part of the dialogue to another
Reason: I don’t know if I’m writing this under the right violation, but one thing that bothered me a little was that if I chose to “show all” in the list view, then went to the map view and back to the list view, all details were hidden again.
Violation 6

Element: The whole system
Violation: 3c. Instructions for use of the system are visible or easily retrievable where appropriate.
Reason: I don’t think the system adheres to this. I cannot find any usage instructions anywhere, so this is not violated by any specific element but by the whole system.

The evaluator does not say whether instructions are needed or not, but since this is entered as a violation, one can assume that the evaluator thinks that instructions are needed.

Violation 7

Element: The link target of the logo
Violation: 4b. Users do not have to wonder whether different words, situations or actions mean the same thing.
Reason: It seemed like the Enea logo and the Map view link pointed to the same view, which I thought was a bit strange. It was not what I expected. I thought the Enea logo would point to information, either about the system, about Enea or some contact information. It was not at all obvious that it was a link to the map view.

Although this design is based on a guideline (5.2), the violation may be a design or implementation issue rather than an invalid guideline. The reason of this confusion may be that the logo is the Enea company logo and not a logo specifically designed for the prototype. Another reason might be that the map view, to which the logo links, might not be the obvious start page of the application. While it is hard to completely disregard this suggested violation, Nielsen’s research backing up that guideline is more substantial than this evaluation.

Violation 8

Element: –
Violation: 8. Good error messages
Reason: I did not see any error messages, so I cannot comment on this one.

Violation 9

Element: –
Violation: 10. Help and documentation
Reason: I cannot find any help or documentation at all.
5.2.5 Evaluator E
Evaluator E states that his overall impression of the application is that it is nice and easy to use.

Violation 1
Element: Map view, “services performed”
Violation: 1a. The user interface is as simplified as possible
           1c. The users' tasks are matched in a natural way
Reason: It’s not entirely obvious that the checkboxes work as logical ANDs (for example plowing AND salting). At a glance, it’s possible that you could see what was plowed OR salted. After a while you see how it works, but you have to try to understand.

Violation 2
Element: List view
Violation: 1b. Navigation is minimized
Reason: Quite a few clicks on “show all”/“hide all” or the “^”/“v” buttons may be needed before you have seen what you want. Also see the comment on 2b below.

Violation 3
Element: The list view button in the menu
Violation: 2a. Words, phrases and concepts are familiar to the user rather than system-oriented
Reason: “List view” could be called “snowplow status” or something similar. The title “list view” does not give users a hint on what that page really contains.

Violation 4
Element: List view
Violation: 2b. Information appears in a natural and logical order
Reason: I would prefer if the list view was organized as a table with one plow per row and a column for every kind of information. The tank levels could preferably be shown graphically. The table could still be filtered the same way.
Appendix C: Evaluation Answers

Violation 5

Element: Filters
Violation: 3b. Users do not have to remember information from one part of the dialogue to another
Reason: When switching between the map view and the list view the choices are reset. You have to select your “filter” again.

Violation 6

Element: Alarm states in the list view
Violation: 4a. Conventions are followed
4b. Users do not have to wonder whether different words, situations or actions mean the same thing
Reason: In the list view the word “Active” is used, both in “currently active”, stating whether the vehicle is running or not and also referring to the alarm state. In the list details the states of the alarm is however “Triggered”/“Enabled”/“Disabled” while the filter buttons are named “Triggered”/“Active”/“Passive”. It would have been better to use the terms “Triggered”/“Enabled”/“Disabled” everywhere.

Violation 7

Element: Map view slider filters
Violation: 5a. The system always keep users inform about what is going on, within reasonable time.
Reason: The map view is updated when the slider controls for “time since” and “speed” are released. It would have been even clearer if the colored routes on the map updated instantly while dragging.

Violation 8

Element: Map view
Violation: 7a. Buttons or other shortcuts are giving access to important or frequently used functions are available where needed
Reason: I miss the ability to find the snowplows in the map (if I accidently would have scrolled the map to Amazonas or Moscow)
Violation 9
Element: Map view, services performed filter
Violation: 10a. Help and documentation is available where necessary
Reason: A small helping information text is needed next to “Services performed”. What can be done? What does it mean if several checkboxes are checked?

5.2.6 Evaluator F
Evaluator F thinks that the interface is great. It looks good and is easy to use.

Violation 1
Element: “Services performed” filter in the map view
Violation: 4. Consistency
Reason: In the list view it’s natural that there is a logical AND relation between the “Currently active” and “Car alarm” filters. I however assumed that the logical relation between the “Services performed” checkboxes was OR. I hence feel that this is unclear, but I don’t know how to make it clearer.

Violation 2
Element: Legend, map view
Violation: 4a. Conventions are followed
Reason: In the map view, when routes are colored by vehicle, it looks like the plows in the legend list are clickable. I don’t know what I would expect to happened, but maybe to see the plow details shown under the list view? This will however probably be strange if “Time since” or “Speed” were selected.

This is an example of the prototype violating one of the guidelines. What is not clickable should not look clickable.

Violation 3
Element: Map view
Violation: 5. Feedback
Reason: It would be interesting to see the snowplows current location.
APPENDIX D: Implementation Notes

The tools and frameworks used when implementing the prototype are briefly described in this appendix.

Development

The prototype was developed using Django on Python 2.6. The coding was done using Eclipse and the Eclipse extension Pydev. A local MySQL database was used, but a remote database works just as well. If a Django application is going to connect to a MySQL database, the Python interface to MySQL called MySQLdb must be installed.

Django sites come with a maintenance script, named manage.py, which among other things can run a development server, generate new Django sites, and create (but not modify) database tables according to model specifications. The database structure is created using the following command:

```
manage.py syncdb
```

To run the development server, which automatically reloads when files are modified, run the following command:

```
manage.py runserver
```

The Django documentation, located at docs.djangoproject.com, has a great getting started guide.

Deployment

Deployment on Apache can be done using mod_wsgi. This requires Apache to have mod_wsgi installed and to be configured to call Django. The following Apache configurations handle all calls using Django:

```
# Setup mod_wsgi
WSGIScriptAlias / /path/to/django.wsgi
```

Note that Django does not handle static files, such as images, styles and scripts. Those files should instead be handled directly by Apache, using an alias:

```
# Application static media files
Alias /static /path/to/django_sites/site/static
```
<Directory /path/to/django_sites/site/static>
    Order Deny,Allow
    Allow from All
</Directory>

If the Django Admin interface is used, an alias for the static files belonging to that must also be added.

The `django.wsgi` file, which starts the Django application, can look like this:

```python
import sys
# Add site path to the Python Path
sys.path.append('/path/to/django_sites/site')
# Sometimes the parent directory must also be added
#sys.path.append('/path/to/django_sites')

# Tell Django which site to use.
# The site name is the name of the directory containing the site.
os.environ['DJANGO_SETTINGS_MODULE'] = 'site_name.settings'

import django.core.handlers.wsgi

# Start the application
application = django.core.handlers.wsgi.WSGIHandler()
```

When deployed, the Django setting called `DEBUG` should be set to `False` in order to use the non-debug versions of error messages.

---

**Frameworks**

**Server-Side**

The server-side code of the application is based on Django 1.1. Django is a Python web framework adhering to the DRY principle. It has an architecture similar to the MVC pattern, but according to the developers’ interpretation of MVC, Django is a “MTV” framework – that is, “model”, “template” and “view”. Models are data objects, and they contain their definition of their database mappings. Views describe which data to show users and templates describes how data is presented. The Django documentation, located at docs.djangoproject.com, is well written and everything you need to get started.

Like most web frameworks, Django comes with an O/R mapper supporting many different databases and several kinds of operations. Django also comes with a template rendering engine, with its own extendable template language.
A Django project is called a “site” and one site can contain one or more “pluggable”/re-usable Django apps. The URL patterns for a site are defined in urls.py, pointing to views in views.py which renders responses based on templates in any of the site’s templates directories. Django has support for form generation and validation, but this is not used in the public parts of the prototype. Site settings, like database access credentials, are set in settings.py.

**Client-Side**

jQuery 1.4 is used to aid various JavaScript functionality where Ajax communication is the biggest part. Documentation can be found at docs.jquery.com.

jQuery UI 1.8 for user interface widgets, mainly the sliders. jQuery UI depends on jQuery. Documentation can be found at docs.jqueryui.com.

YUI CSS Reset 2.8 by Yahoo is used to reset the default styles so that they look the same on more browsers.
Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

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

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

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