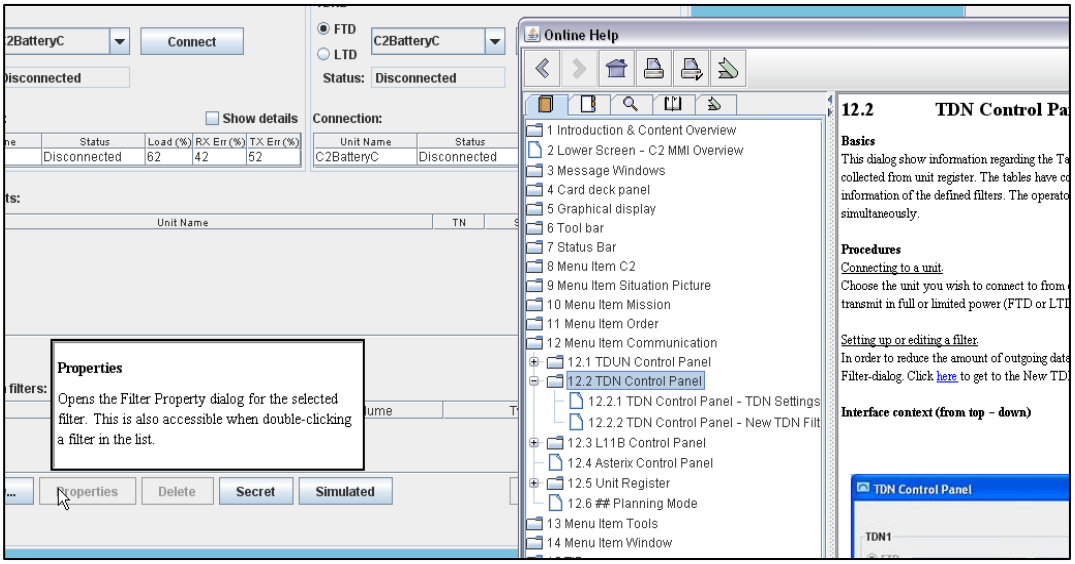
CHALMERS



Online Help

The implementation of user manuals into a radar application from Saab Microwave Systems

Master of Science Thesis in the Programme Interaction Design

Sebastian Dreyer

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, November 2009

Online Help
The implementation of user manuals into a radar application from Saab Microwave Systems

SEBASTIAN DREYER

Examiner: OLOF TORGERSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

The picture on the cover page is a screenshot from the final prototype showing a window with a Pop-up, explaining the button the mouse pointer is hovering over, and to the right is the online manual (help content) with the topic explaining the window with the Pop-up.

Department of Computer Science and Engineering
Göteborg, Sweden November 2009

Abstract

This thesis was done at Saab Microwave Systems, a business unit at Saab AB, and focused on how Online Help (OH) could be implemented in one of their products, as well as if it was possible to use the already existing documentation to create the help content to the OH without having to rewrite it all.

Suitable OH platforms were researched and tested. Afterwards it was decided to use JavaHelp because it was the most versatile as well as supported *Pop-ups*.

The development process was done iteratively with three prototypes. Between each phase a usability test was carried out, and the result was used in the creation of the following prototype. The third prototype included a fully implemented OH system with online manual, F1 key support to access it and Pop-up help in the whole application. The third test let the testers complete an assignment by either using the traditional manuals or the OH, in order to see if there was an advantage to use one over another.

The conclusion is that it is possible to, cheaply and relatively easy, implement OH in an already finished application. Although JavaHelp is far from a perfect solution and need more work if it is going to be used commercially. It is also possible to create online manuals from already existing documentation such as Framemaker or Word documents by using RoboHelp.

Sammanfattning

Det här examensarbetet gjordes på Saab Microwave Systems, en affärsenhet inom Saab AB, och handlade om hur Online Hjälp (OH) kunde bli implementerad i en av deras produkter, men även att ta reda på om redan existerande dokumentation kunde användas för att skapa hjälp materialet till OH utan att behöva skriva om allt.

Lämpliga OH plattformar undersöktes och testades, i slutändan bestämdes det att JavaHelp skulle användas pga dess mångsidighet och att det stödde *Pop-ups*.

Utvecklingsprocessen gjordes iterativt med tre prototyper. Mellan varje fas så gjordes ett användningstest vars resultat sedan användes i utvecklingen av nästa prototyp. Den tredje prototypen innefattade en fullständig OH med online manual, F1 support för att använda den, och Pop-up hjälp i hela applikationen. Det tredje testet lät testarna lösa en uppgift mha antingen pappersmanualerna eller OH, för att se om det var en fördel att använda den ena hjälp typen utöver den andra.

Slutsatsen är att det är möjligt att, billigt och relativt enkelt, implementera OH i en redan färdig produkt. Dock så är JavaHelp långt från en perfekt lösning och kommer att behöva arbetas om för att kunna användas kommersiellt. Det är också möjligt att skapa en online manual från redan existerande material som Framemaker och Word dokument genom att använda programmet RoboHelp.

Preface

This thesis is the last step in my education in Interaction Design at Chalmers University of Technology and the IT University in Gothenburg. The thesis took place at Saab Microwave Systems (SMW) in Gothenburg during May-October, 2009.

I would like to thank my supervisor/tutor Ulf Dahlgren at SMW for giving the technical support I needed to do my prototypes. Gustav Klock that as the resident Human Factor Engineer at SMW help me a lot with interaction design related issues and acted as an unofficial second supervisor/mentor, it was great to have someone to talk ideas with. Rafael Aramburo, my boss, for helping me to get this thesis. Also like to thank everyone else at SMW for acting as test users or otherwise helping me with this thesis and making me feel welcome.

I would like to thank my supervisor at Chalmers/IT University Staffan Björk for keeping me on track and not lose the sight of the goal of this thesis, as well as being an invaluable support at writing this report.

Contents

1 Introduction

This report will describe a thesis project done at the company Saab Microwave Systems (SMW) located in Gothenburg, Sweden.

Saab Microwave Systems develops sensor systems that are used for military purposes around the world. In order to use these sensor systems different kind of software is also developed. The software is often custom order and are only developed and sold to specific customers. The software is then delivered to the customer together with thorough documentation describing how the software works, but the actual software has no kind of help system in it, only traditional manuals are available. The main reason for this is that it would take to long and cost too much to implement, considering the limited customer base. Another lesser reason is that the people that are going to use the software have been trained to use these systems and thus not need a help system.

Recently this has started to change and a need for some sort of digital help other than the traditional manual. The foremost reason to this is to improve the operator's operation environment by giving him quick access to the operations and controls manuals in the form of Online Help. This is one step to, in the future, stop shipping and making the part of the complete documentation that is classed as the user manual, although as a first step it is important to be able to generate the Online Help and traditional manual from the same source and offer this to the customers.

The introduction of an Online Help system into SMW's products can be the start of introducing an iterative way of developing help system/manual that will result in a better product as well as reduced development costs.

This thesis is a first step to achieve these goals.

1.1 Purpose

The purpose of this thesis is the following:

> *Implement a prototype Online Help system in an already existing Saab product and determine the possibility of using already existing documentation to generate the content of the Online Help.*

This prototype will show how an Online Help system would look and work and thus give SMW a ground to stand on in future development of it. To be able to do this a number of other problems have to be solved first. First and foremost some background research has to be done on what Online Help is. But more importantly how can it be made *useful* and *usable* and what is expected from it? Another important thing is that since SMW wants to get rid of the traditional user manual in benefit of the Online Help they to convince their customers that this is the way to go. This should be taken into consideration while doing the thesis.

A runtime platform has to be found and tested in order to implement Online Help, as well as finding a suitable program to generate the Online Help content from the existing documentation. There are a number of platforms that can be used, but it is important

that the platform will work in a Linux/Java environment, because it is this environment that SMW develops their software in.

The reason it is important to use the already existing documentation to generate the content of the Online Help is because SMW wants to avoid as much extra work as possible since they have limited time and resources to put on this. As long as the traditional manual is requested by the customers it is important to generate it from the same source as the Online Help. One complication/possibility is that the department that handles documentation at SMW today develops an IETM (Interactive Electronic Technical Manual) based the system ActiViewer, and the material is therefore adapted to ActiViewer. Optimally SMW wants to use the same material to create the traditional manual, IETM and Online Help.

1.2 Constraints

As stated above the point of this thesis is not to implement a complete working help system but rather explain how such a system would work and how to implement it. The prototype will have working context-sensitive help for a number of windows (those that is deemed necessary in order to demonstrate the help system) as well as online manual in the product. It will not have help for all windows, and the online manual will feature only part of the user manual. The importance is to show how to use suitable tools to create help system files from the source material.

1.3 Saab Microwave Systems

Saab Microwave Systems is a leading supplier of Radar Systems that are used for military purposes around the world. They have done this for over 50 years and are considered the world leading competence center for microwave and antenna technology. So far over 3000 sensor systems have been delivered to more than 30 countries. [Saab 2007] These systems are made to function in a variety of conditions and environments, no matter if it's in the air, on land or at sea. Examples of these systems are ERIEVE (air/sea), GIRAFFE AMB (land), and Sea GIRAFFE AMB (sea). [Saab 2006]

Special software is needed to use these Radar Systems. The software also differs from system to system. The system this thesis will focus on is the GIRAFFE AMD (GAMB). To this system there are two software products that work together, one Sensor Graphical User Interface (SGUI) that handles and show the sensor input from the local connected GAMB and one Tactical GUI (TGUI) that show the sensor input from the local connected GAMB and all other C2 (Command & Control) Units (see Figure 1-1) and are used to give orders and other tactical functions. Only the latter one is included in this thesis.

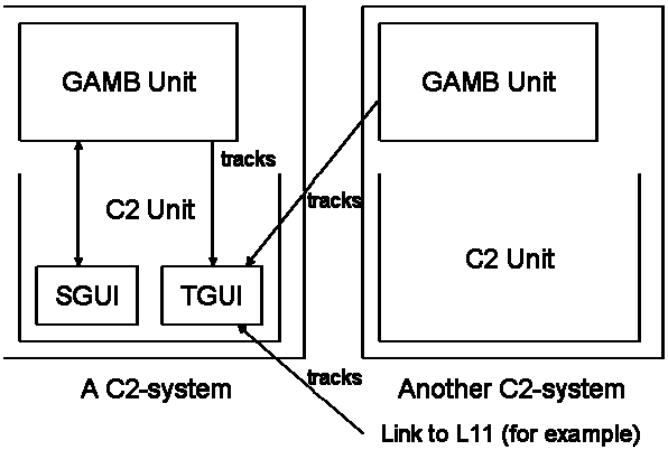


Figure 1-1: A C2 System

1.4 Outline

The disposition of the rest of the report is as follows:

A background chapter describing how everything was before this thesis started, i.e. the software that the Online Help was going to be implemented on, the existing documentation, practices that are currently being used as well as those that are about to be introduced, earlier attempts at Online Help, and descriptions of platforms that can be used to implement Online Help.

Next is a theory chapter explaining User-centered Design and *usability*, followed by a chapter describing the methods used in this thesis. The original planning of the thesis is explained in the chapter after that. This is followed by a small chapter explaining the design guidelines chosen. The mock-up chapter explains the mock-up made in the beginning to illustrate how the thesis should look and work. Afterwards there is the platform search chapter explaining how the different Online Help platforms were found and tested.

The next three chapters focus on the three prototypes made and the test performed on them, followed by a test result analysis chapter explaining the test results from the final usability test. Next is the result chapter giving a thorough walkthrough of the final prototype.

The discussion brings up different aspects of this thesis as the prototype, development process, this report, test results, future work etc. followed by the conclusion chapter. Lastly there are the reference list and appendices.

2 Background

This chapter describes how SMW is currently developing their products and manages the documentation.

2.1 The Software

The C2 systems as mentioned in the Introduction are divided into two parts, the SGUI and TGUI. The SGUI takes information from the local sensor GAMB unit (see Figure 1-1) to show how the immediate area looks like. This is shown on the upper screen of the operator workstation (see Figure 2-1). The TGUI takes input from the local GAMB as well as input from all other C2 Units connected to the Tactical Network. These inputs are then combined and shown in a graphical display window in the TGUI, which is placed in the lower/middle screen of the workstation. The operator now has a full overview of the whole tactical grid and can plan and give orders on a much greater scale than before.

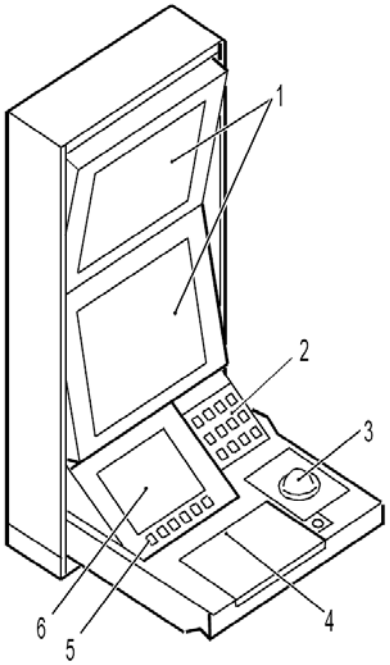


Figure 2-1: The Operator Workstation

2.2 The documentation

As things are right now the documentation for the SMW software is not written until the end of the development cycle. This because that the documentation department (DP/K) are not usually part of the development team and are thus not contacted until the end of the development. The amount of work the DP/K currently is having also affects how quickly they can start working on a specific project.

Making the documentation at those stages have disadvantages, such as the people writing the documentation usually have not worked with the system before, and thus must spend a lot of time figuring out the system and interface, talking to the programmers, and even then the manual probably will not be entirely correct. This as

well as writing manuals from scratch is very cumbersome and also take into consideration that this is done under a limited time period. This more times than not results in manuals that do not explain in enough details the functions and control of the interface.

Other disadvantages are that because the software, and thus the interface, is not complete when the writing of the documentation starts means that the control manual in particular must be updated several times before the software is done. This is a natural thing in development cycle but does make DP/K's work problematic, since the continuous update of the software also forces a continuous and often retroactive update of the documentation.

Worth mentioning is that the employees at DP/K are in constant contact with the customers of SMW's products and therefore have lots of experience of what the customers think, do and like, knowledge that would be priceless if they would take an active part of the software development.

2.3 New practices

SMW is currently working on changing the way software and documentation is done. Although this is still in the early stages and will probably take some years before it is fully established. One of the main goals is to try to focus the development on *Usability* (see chapter 3). To help achieve this goal SMW recently hired a Human Factors Engineer (HFE) that will help the programmers to obtain enough resources to make the interface *usable*, using their full potential as interaction designers and usability testers. This is also something that Saab as a whole is interested in, so there are usually a few meetings a year with people working with HFE and interaction design from all over Saab discussing this. The introduction of Online Help into SMW's products is one of these goals.

Currently there is a person that documents all that happens in the interface, which DP/K then uses to base their work on, like manuals. There is talk about trying to continuously update the documentation with the software during development. The thought is that one of the employees at DP/K will be a part of the development team and help with writing the Online Help and user manual, i.e. the source material to these two. At the end of an iteration the Online Help will be tested/verified which will hopefully increase the quality of the documentation. In addition the rest of the team could use the Online Help to learn more about the system. Small changes made in the interface can be noted and implemented into the documentation by the constructors. The Online Help will thus be continuously be inspected by the entire team.

2.4 Earlier attempts with Online Help

This thesis is not the first attempt that SMW has made with Online Help, a few years ago an attempt to incorporate Online Help into one of SMW's products using JavaHelp and RoboHelp (a publishing tool to create help content). This was a part of a customer project there they wanted to test how Online Help could be used in an application. This was never meant to be more than a quick test, because of tight deadlines and limited budget. The test showed that it was possible to implement Online Help but due the limited time and budget it was not feasible to put extra resources to do this at that time.

2.5 Other applications with Online Help

There are of course other applications by different companies that have Online Help. Mostly though they are limited to just implementing help content/online manual in their applications. This include popular applications like Microsoft Word, or Eclipse which has a online version of their help content which demands an internet connection to use. There are very few applications that have context-sensitive help, *Pop-up* help for example. No such applications were found and studied during this thesis.

2.6 The Available Online Help Platforms

There are a number of available platforms on the market that allows implementation of Online Help in the product. This thesis will only focus on those that are Java based, since that is the programming language used in the application. There are three platforms available freely on the market, and two more that are Saab solutions. These are Eclipse Platform Help System (EPHS), JavaHelp, Oracle Help, and within Saab, ActiViewer and a Firefox Portable solution.

2.6.1 Eclipse Platform Help System

In Eclipse it is possible to create help content (Online Help) to a project in the form of a plug-in. This is easily done by creating a new project and selecting 'Plug-in Project' instead of 'Java Project', after filling out some standard information it is possible to choose a 'Help Content' template that creates a table of content among other things. The table of contents is located in a XML file called *toc.xml* that contains labels and anchor ids to each topic. The topics are also saved in XML files and links to an anchor id in *toc.xml*, they also contain the structure of the topic, like main and sub topics, linking to the HTML (or XHTML) files containing the actual content. The created help content is easily tested by opening Eclipse Help Content window, the new help files should be at the bottom of the table of content. [Eclipse 2009]

2.6.2 JavaHelp

JavaHelp makes it possible to implement Online Help into a Java project. JavaHelp is not included in the JDK or JRE, it must be downloaded separately. The implementation can be divided into two separate parts, creating the help files needed for the Online Help and implementing it into the code. [JavaHelp 2004]

The Help files

There are three important files (not counting the HTML topic files) that are needed to make the help content work. The main one is the *helpset* file which is read by the application as soon as the JavaHelp system is activated. This file defines the help content window for the application and contains a set of data that comprises the help system. It contains information where the so called *map* file is, view information that describes what types of navigators that are being used in the help content window; standard navigators are table of content, index and full-text search. Other information is what title shown in the top of the help content window, as well as presentation settings, i.e. the size of the window, if there is a toolbar or not and if so what it will contain, like *print* and *home*.

The second important file is the *map* file mentioned above. This file is used to associate topic IDs with the URL or path name of the HTML topic files. Lastly there is the *Table*

of Contents (TOC) file that describes the TOC navigator and layout of the TOC. The format of the TOC file is based on the World Wide Web Consortium (W3C) Extended Markup Language (XML). The helpset and map files also uses the same format.

Besides these three there are other optional files that can be included, these are the *index* file, *glossary navigation* file, and *favorites navigation* file. All the help files can be compressed into a JAR (Java Archive) file, it is not necessary to unJAR the files to run the help system.

It is also possible to merge two or more helpsets. This is useful when installing different modules for a platform, each module has its own helpset that will be merged with something called the *master* helpset, probably the helpset of the main application or similar. There are two ways of merging helpsets; one is static merge which is done by specifying the helpsets in the XML code of the master helpset's helpset file. The other is done by writing code in a Java program that uses JavaHelp's API, this is called dynamic merge. Which of them is used depends on the structure of the application. [JavaHelp 2004]

Implementing JavaHelp into the code
To be able to use the help content in the application, there are two important things that have to be done. The first thing is to include *jhall.jar*, which contains all the classes that JavaHelp uses, in the project. The other one is to create a reference to the helpset file so the application can access the help content.

To enable context-sensitive help for windows and components in the application, they must be linked to a topic ID that exists in the *map* file. This can be done in several ways in JavaHelp, but the most common one is to use the function `enableHelpKey()` to enable help when pressing the F1 key, it recommended to use this on windows. If F1 is not going to be used but the component needs a topic ID anyway, when using Pop-up help for example, the function `enableHelp()` can be used instead. [JavaHelp 2004]

Using the `enableHelpKey()` function is all that is needed to get help by pressing F1. Making Popup help work needs a little more work. First of all a `mouseListener` must be added to the component that needs Pop-up help. When a component with an added `mouseListener` is activated the responding code will create a Pop-up menu that shows the HTML file that is linked to the topic ID of the component.

There are other ways of using JavaHelp than those mentioned above, help buttons for example instead of using F1, but since these were not used in this thesis it is no real reason to describe them here.

2.6.3 Oracle Help

Oracle Help is very similar to JavaHelp, which is only natural since they have used JavaHelp as a base for their own help system. First of all the help content window has a more modern look to it (see Figure 2-2) and it is possible to undock the topic window from the navigation window to get two different windows (see Figure 2-3). The benefit of this is not clear. [Oracle 2009]

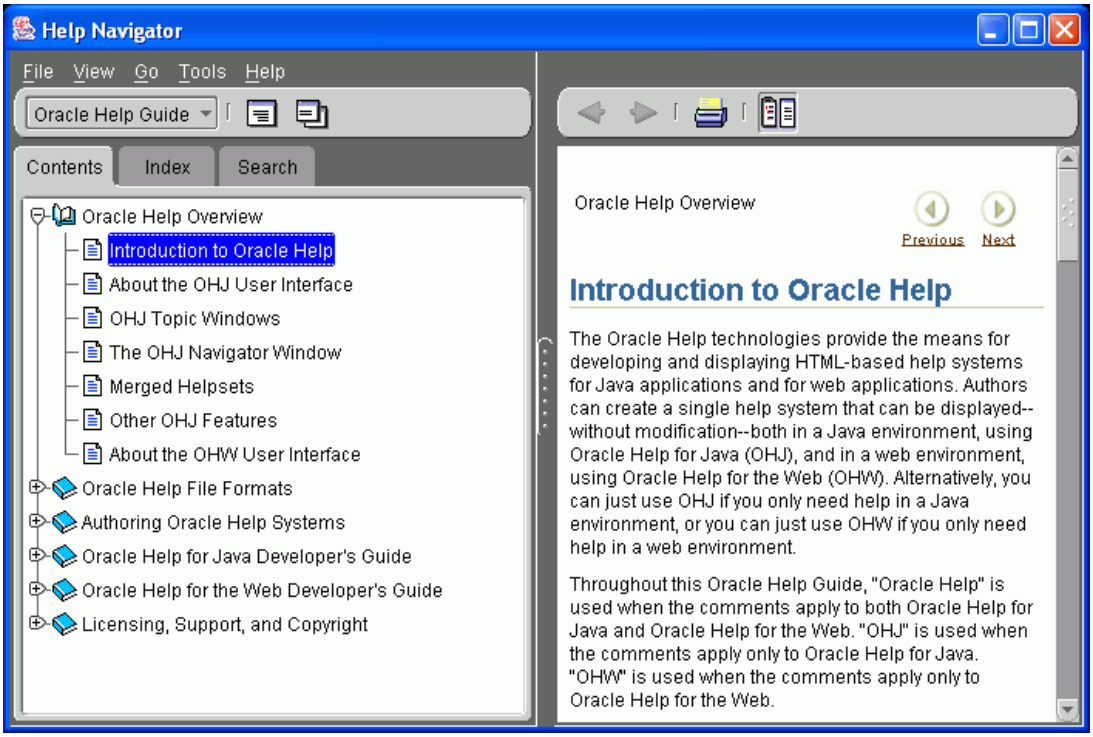


Figure 2-2: Oracle Help Content Window Docked [Oracle 2009]

Oracle Help like JavaHelp also supports merging of helpsets. This is done in runtime in Oracle Help, making it possible for multiple authors to create multiple helpsets that can be merged seamlessly without having to rework the system. The table of contents from the different helpsets will be put atop of one another, except nodes with the same text and topic ID; these are combined into one, as long as there are no conflicts.

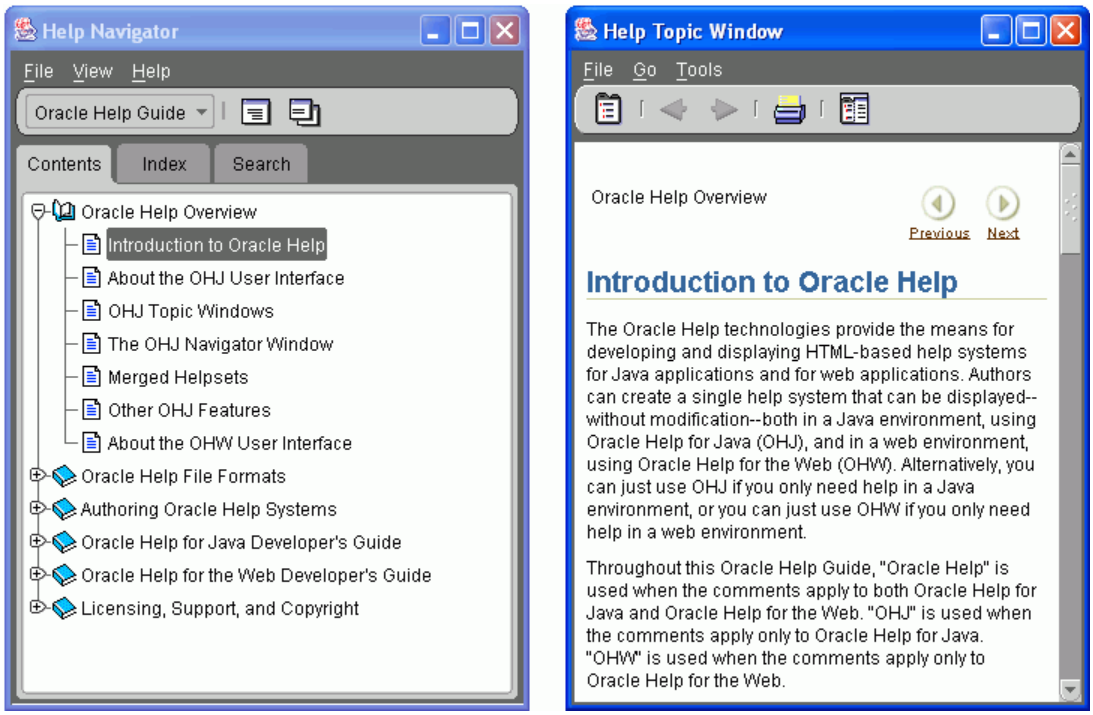


Figure 2-3: Oracle Help Content Window Undocked [Oracle 2009]

The help files are the same as JavaHelp, there is the *helpset*, *map*, *TOC*, *keyword index*, and *search index* file, the new one that is not in JavaHelp is the *link* file. The helpset file is more or less the same as JavaHelp except that *presentation* is called *wintype* instead and there is a new option called *link* that have to do with the *link* file mentioned. A few internal differences and some new options, otherwise it is the same. No change in the *map*, *TOC* or *index* file. The *link* file is an XML file that defines link IDs and associates them to several topic IDs, i.e. making it possible to associate an HTML link with several targets letting the user choose which of these to follow.

It is also possible to use Oracle Help for the web, which is very similar to Oracle Help for Java, but this is not part of this thesis and thus not pursued any further.

To use Oracle Help in the code the three JAR files *help4*.jar, *ohj-jewt.jar* and *oracle_ice.jar* must be added to the project, and then a reference to the helpset must be created much in the same way as JavaHelp. After that it is a simple thing to add Online Help to components by using the function **`addComponent(Component` *`component`*`, String` *`topicId`*`)`**. After this a context menu will be opened if a user right-click on such components, and doing this will launch the help content. It is also possible use F1 to do this, but in order to make this work another version of the **`addComponent()`** function has to be used, in that function the boolean parameter **`needF1Help`** must be set to *true* first. [Oracle 2009]

2.6.4 ActiViewer

ActiViewer is a program that the documentation department (DP/K) at SMW is using to run their IETMs (Interactive Electronic Technical Manuals). IETMs are the electronic manuals that are currently shipped to customers as a complement to the traditional manuals. It was a request from DP/K that ActiViewer should be used in this thesis if possible to show the help contents in the application.

The manual is divided into a number of XML files that ActiViewer uses to build the IETM. The IETM looks and works similar as the normal help content (see Figure 2-4).

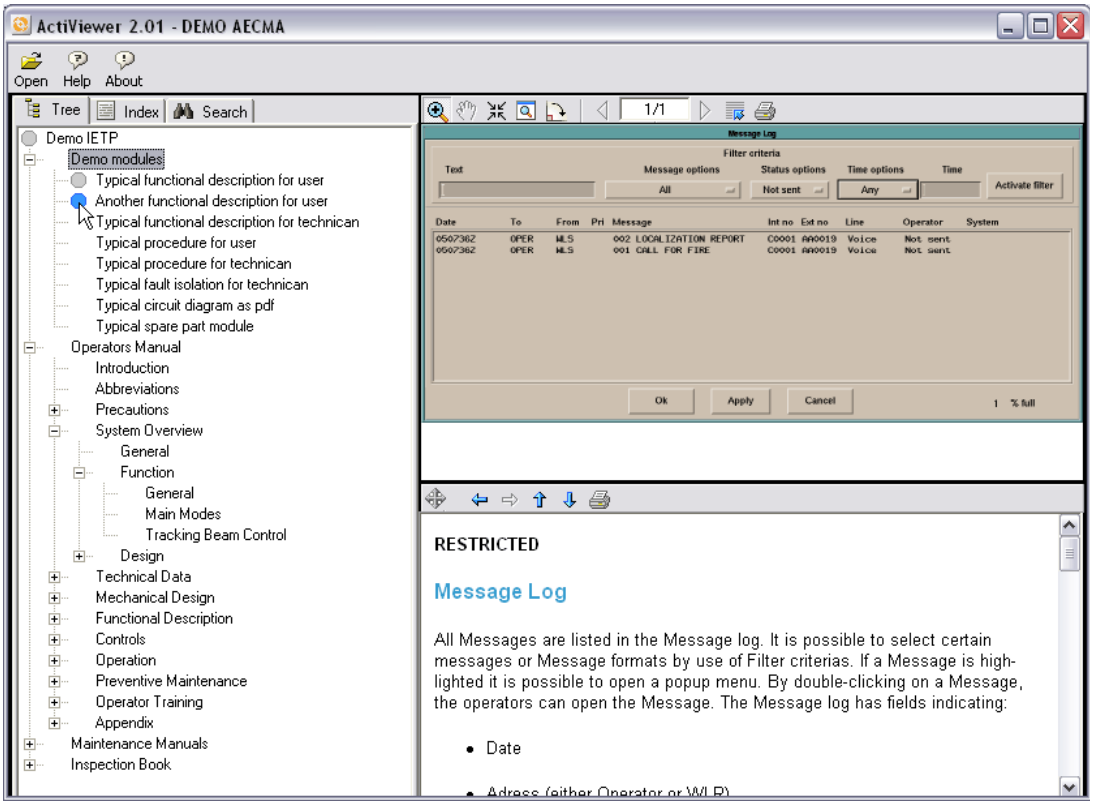


Figure 2-4: ActiViewer IETM

2.6.5 Firefox Portable solution

It was discovered while researching the different help system platforms that another company within the Saab Group, Saab Systems, had developed their own version of Online Help. This version was based on Firefox Portable and a XML Client. Their reasoning for using Firefox Portable as their runtime environment was that they would not need to install it to be able to use it; it can easily run on a USB stick or even a CD. This is important because most customers do not allow more than the main software to be installed on their computers. Also the software has to be able to run in a UNIX environment and Firefox are thus more or less the only web browser to use.

Worth mentioning is that before they started working with Firefox Portable they tried to use JavaHelp, but they found out that the browser that JavaHelp uses was not up for the task. When the HTML file becomes too large the browser becomes sluggish and the response time increases rapidly. This was five years ago but considering that not much development has been done on JavaHelp recently this problem may still be there.

Saab Systems uses a stripped version of Firefox Portable which has had many of its functions removed as well as most of the interface, such as toolbars, menus and address field (see Figure 2-5). The structure of the Online Help consisted of XML files and all functionality is handled by a Java-script, which makes sure that the users cannot access the file system of the workstation in any way.

Everything is run from a start file which uses an index file containing about 70000 words or 300-400 documents in both XHTML and PDF. It is easy to search in. The Online Help can be started in several ways, one is to use the start file directly or by

right clicking on a component, a “What is this?” button will then be shown that the user can click on to open the Online Help. The button is greyed out if there is no information about the component. Each such component has an URL linked to it so when the “What is this?” button is clicked Firefox will show the topic of that URL. The URLs are ordered hierarchy depending on how the components are placed in the GUI.

Each night a XML file is generated containing the changes that have been made during the day, so that the developers easily can implement Online Help to new buttons and such. This makes the system easy to use and reduces the workload on the developers.

They work in a UNIX environment, while SMW develops products in a Windows environment that are meant for UNIX environments. Because of this it is unlikely that the developing tools they use will work in Windows.

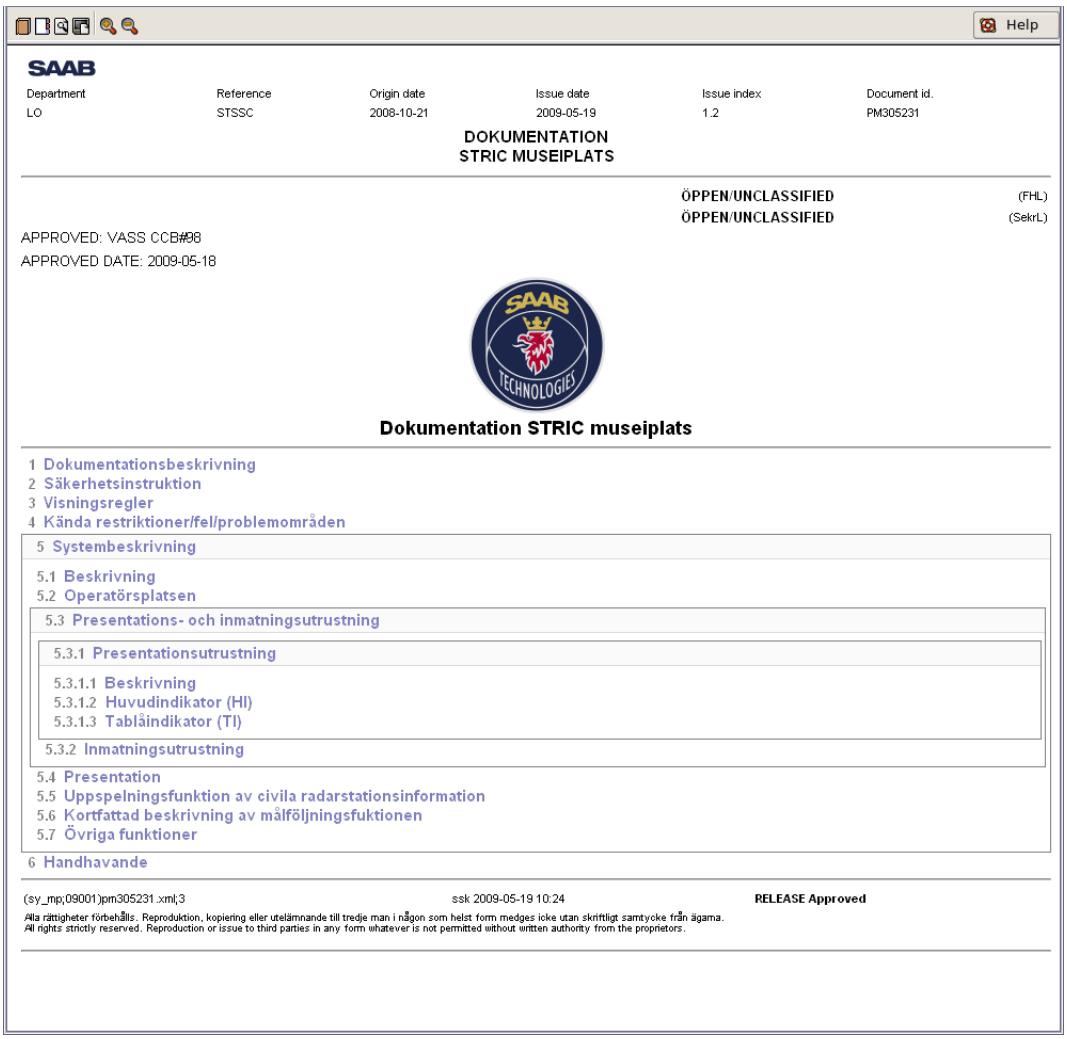


Figure 2-5: Online Help with Firefox Portable

3 Theory

Since this is a master thesis in interaction design it means that *Usability* and *User-Centered Design* is a main part of the thesis. There are also a number of methods that can be used in an interaction design process, whereas some of them will be describes in the next chapter.

3.1 Usability

Today, the every day definition of the term *Usability* can be considered highly diverse. There are often several unknown aspects influencing people's interpretation of the term, such as what user profile the term is supposed to reflect as well as the system complexity in which the user is supposed to operate in. The ISO-definition says:

> *"The extent to which a product can be used by specified users to achieve specified goals with **effectiveness**, **efficiency** and **satisfaction** in a specified context of use."* [ISO 9241-11]

But what do effectiveness, efficiency and satisfaction mean in this context? ISO-DIS 9241-11 states the following:

> *"**Effectiveness**: The accuracy and completeness which users achieve specific goals.*
> ***Efficiency**: The resources expended in relation to the accuracy and completeness which users achieve specific goals.*
> ***Satisfaction**: Freedom from discomfort, and positive attitudes towards the use of the product."* [ISO 9241-11]

ISO-DIS 9241-11 is a definition of how to identify information that is necessary when specifying or evaluating *Usability* of a visual display terminal by measuring the user performance and satisfaction. It is a guide that describes general principles and techniques, rather than requirements, and can be used in procurement, design, development, evaluation, and communication of information about *Usability*. [ISO 9241-11]
This is one explanation, one that is seen as an international standard nevertheless, but as mentioned above, there can be other ways to explain/describe *Usability*.

Usability is also defined as a *quality attribute* in interactive products, meaning that products with high usability fulfill the customers' and target groups' (groups of users that have similar expectations and intensions of using the product) purposes.

To create a usable product three important aspects have to be taken into consideration:

- *The human system*: general and specific skills that the users have.
- *The context in which the product is going to be used*. The product has to be adjusted to physical, psychological, social, and organizational contexts that it is going to be used in.
- *The benefit that is expected from the product*. Both the one that uses as well as the one that provides the interactive product expect to benefit from it. The provider may expect economical gain while the user expects efficiency and/or satisfaction.

Usability is not something objectively observable nor does it have internal product attributes like a color or function. Usability is something that emerges from product use. *Quality-in-use* is a concept that puts the focus on the use of the product instead of the actual product. This makes it clear that usability is a measure of quality and relies on the context in which it is used. [Ottersten et al 2008]

Ottersten & Berndtsson (2008) also gives their own view on ISO 9241-11. They think that the ISO definition puts to much focus on that the target groups' needs are fulfilled. An equal amount of effort should be put in making the buyers' needs and purposes fulfilled as well in order to call a product usable. The ISO definition and much of the literature within this area seems to assume that the buyer already have determined the products purpose before any development have been done.

3.2 User-Centered Design

User-Centered Design (UCD) is described differently depending on who is asked, but there is an ISO standard describing it [ISO 13407:1999]. According to this ISO standard UCD is a multi-disciplinary activity. It incorporates human factors and ergonomic knowledge and techniques with the objective of enhancing effectiveness and productivity, improving human working conditions as well as counteracting the adverse effects of use on human health, safety and performance.

There are four iterative steps of UCD that are commonly used:

- Understand and specify context of use
- Specify the user and organizational requirements
- Produce design solutions
- Evaluate design against requirement

These are illustrated in Figure 3-1 below. The process is supposed to be iterated until the objectives are satisfied. How these sequences are performed or the level of detail and effort used depends on the design environment and the stage of the design process.

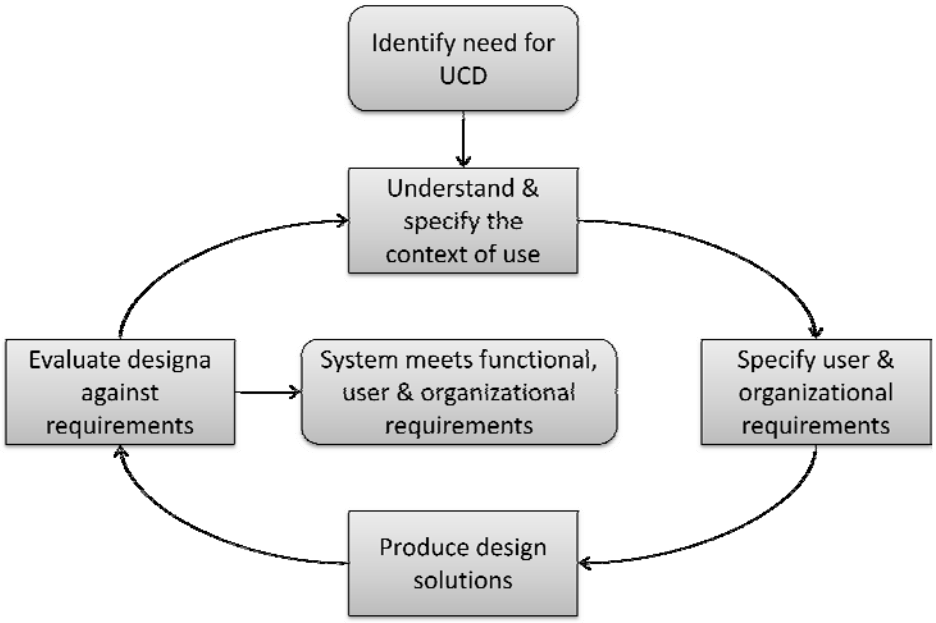


Figure 3-1: An illustration of the UCD iterative process

The purpose of the UCD process is to make the product easier to use, improve user satisfaction and reduce comfort and stress, improve the productivity of users and operational efficiency of organizations as well as improve product quality, appeal to the users and provide a competitive advantage. [ISO 13407:1999]

But as the name UCD states the point of ISO 13407 is to stress the user's role in the development. Cooper (2007) has made his own interpretation of how this standard should be followed and puts his focus on the goals of the users. This Goal-Directed Design is described in the next chapter.

3.2.1 Goal-Directed Design

Goal-Directed Design (GDD) [Cooper 2007] is the process of putting the needs (or goals) of the users as the main focus of the project, so that they will be satisfied, effective, and happy to use that product and therefore willing to open up their wallets and buy it and recommend others to do the same.

This sounds like an easy thing to do, to design a product that users enjoy using. Unfortunately it is not that easy, and the reasons for that are many. In the center of it all is the tug-of-war between the developers and marketers. The marketers are good at understanding the market and position of products within that market; their involvement in the actual development is usually the delivery of a requirement list. These requirements often have little to do with the users' needs or desires and more to do with guesses based on market surveys, what users say they want. The problem here is that users seldom know what they want. Developers on the other hand focuses more on solving technical problems, follow good engineering practices, and meet deadlines, as well as receiving conflicting and/or confusing instructions that they have to follow, rather than consider on how the users will think and use the product. Any real interaction design choices are made when the product is almost done thus making it near impossible to make any real changes in it. The result of this is products that irritate, reduced productivity and otherwise fail to meet the users' needs.

Another problem is that products are designed in way that makes them seem rude, blaming users for mistakes that is not their faults and then agreeing to this by pressing *OK* (see Figure 3-2 for an example). They have a tendency to interrogate the user by asking question like "Are you sure you want to delete this file?" or "Do you really want to empty the recycle bin?" Applications also require users to think like computers. To rename a document for example, the user either has to use the *Save As* menu command to create a new file and then delete the old one or close the document and then rename it, there is no *Rename* menu command. This is the natural way for a computer to work but not how a normal person does things.

Figure 3-2: A non-helpful warning window [Cooper 2007]

The main problem is that the digital industry does not have a good understanding of what makes users happy. Sure they have information on what kind of market they are in and what jobs they have, how much money they make and spend and what they like to buy. But this does not really tell anything about what makes them happy or how they are going to use a product. There is also conflicting interest between the users and the ones that build the products, the programmers. Programmers often have to choose between making the coding easy for themselves and making the product easy to use. It is not hard to figure out which of these tracks they are going to take. It is never a good idea to let the people who are building the product also design it. But the real issue here is the lack of any process to create successful products. Lacking may not be the right word, there are different processes used by both by the engineering and marketing departments, but they are focused on other things than making products that meets the users' needs and desires. [Cooper 2007]

Designing Digital Products

It is important to realize that creating a digital product is not the same thing as creating a chair or a car. Digital products are so much more focused on the interaction than most physical products. The traditional design used by the industry is therefore of little use when designing digital products. This new kind of design, interaction design, requires an understanding of the user's relationship with the product from before purchase to end-of-life. The field that has most in common with interaction design is that of architecture. Architects need to understand how humans that occupies a structure lives and work, to be able to design buildings that support and facilitates those behaviors. This is very similar to digital products, the interaction designers also need to understand how the humans using their product lives and work to be able to design a product that support and facilitates those behaviors. The difference is that the field of architecture is a very old and well-established while interaction design is new.

In order to address this kind of new behavior-oriented design one has to understand the goals each user has. But what are user goals? How are they identified? Are they same for all users? These questions among others need to be answered in order to do this. The goals are not always what they seem; an accounting clerk's goals for example may seem to process invoices efficiently. This is rather his employer's goal not his, his goal is probably to appear competent at his job and trying to be concentrated at his work while performing routine and repetitive tasks, even though he is probably not even consciously aware of this.

This means that products built with business goals in mind alone will eventually fail, users' goals needs to be addressed for product to be successive. Many applications today fail to do this; their interfaces makes the users feel stupid, causes them to make mistakes, require to much effort to operate efficiently, and importantly they don't provide an engaging or enjoyable experience. The companies' priorities are wrong. They focus too much on implementation issues rather than the needs of the users. Even when they focus more on the users it is not always they can do anything to change the product because the conventional development process says that the coding is done first and designing the interface done later. This gives little room for changes. If this is not the issue then the developers tend to pay too much attention to the tasks the users are doing rather than the goals they are performing to do those tasks. It is worth noting that goals are not the same thing as activities or tasks. Goals are an outcome that is reached by doing tasks and activities.

There is often a misconception that ease-of-use should be prioritized when designing a product, this is an important guideline but following rules that are disconnected from user goals are not good design. For example, people using an automatic call-distribution system do not want to go through a step by step call-routing process each time they route a call, they want a system that can efficiently route calls and rapidly complete them. Here ease-of-use is of less importance. But there are systems where easy-of-use is a major goal for the users, for example kiosk at museums or similar. [Cooper 2007]

The GDD Process

Most companies do not use a good enough process to design a digital product, if they even have such a process. The problem is that the word design has lost its meaning in the industry, nowadays it is more a word describing the looks of the interface not its function. For the word to have its meaning back the designers must have a broader role than they previous had. Today the roles in the companies are too precise, researchers researches the market and designers do design. To be able effectively and systematically translate the research into a detailed design specification designers have to be let into the loop and take part of the knowledge firsthand to better understand the users.

GDD combines several techniques like ethnography, stakeholder interest, market research, detailed user models, among others. This defines a process that can be divided into six phases: Research, Modeling, Requirement Definition, Framework Definition, Refinement, and Support (see Figure 3-3).

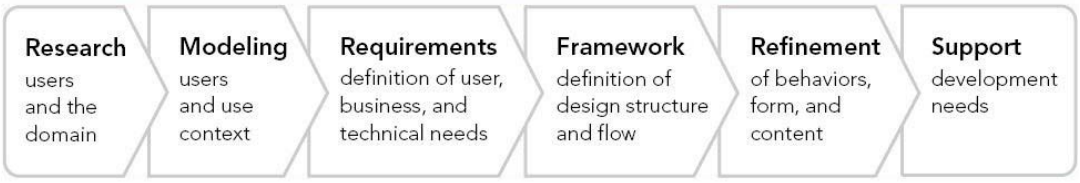


Figure 3-3: The Goal Directed Design Process [Cooper 2007]

The research phase uses different ethnographic techniques to provide qualitative data about users of the product. This results in behavior patterns that suggest goals and motivations that the users have in order to use the product. These can be used later on in the Model phase to create personas. Personas, or user models, are detailed user archetypes that represent groupings of behaviors, attitudes, aptitudes, goals and motivations discovered during the research phase. These are used in several of the other phases, like main characters in scenario-based approaches to generate design concepts in the Framework Definition, provide feedback that enforces design coherence in the Refinement phase, among other things.

The Requirement Definition uses scenario-based design methods, focusing on meeting the goals and needs of specific user personas. Each persona is analyzed through an iteratively refined context scenario that starts with a "day in the life" of the persona using the product, describing high-level product touch points, and then successively defining detail at ever deepening points. The result of this is a requirement definition that balances user, business, and technical requirements that the designers can follow.

During the Framework Definition phase the overall concept is created, defining the framework for the product's behavior, visual design, and possibly physical form. An interaction framework is then synthesized by using two methodological tools together with the context scenarios. These two are sets of interaction design principles and interaction design patterns. The first one is used to help determine suitable system behavior in different context, and the other one contains solutions to previous analyzed problems. These patterns are not set in stone but continue to evolve as new contexts arise and thus provide help with proven design knowledge when approaching difficult problems. The interaction framework is then translated into design elements by using interaction design principles and organized into design sketches and behavior descriptions. This will result in an interaction framework definition that provides logical and formal structure for details to come in the Refinement phase. The interaction framework is also used by the interface designers to produce several suggestions for a visual framework, also called visual language strategy.

The Refinement phase is similar to the previous phase but focuses more on details and implementation. Interaction designers focus on task coherence and validation scenarios, while visual designers define the style of the interface, such as sizes, icons, and other visual elements. All this results in a detailed design document delivered in traditional or interactive media.

The Support phase is not entirely a phase per se; it is something that is used during the entire development to help answer developers' questions as they arise. This helps the development team to prioritize their work and make trade-offs to meet deadline. Not

doing this could result in compromises that threaten the integrity of the product's design. [Cooper 2007]

4 Methods

This chapter will describe how the theory from the previous chapter was adapted and used in this thesis.

4.1 Usability testing

Usability testing (also wrongfully called user testing – it is the actual use, not the user, that is being tested) is a collection of techniques that lets developers test their products to measure the characteristics of a user's interaction with a product, with the goal of determining the usability of that product. Usually the users are put through standardized test that they have to solve to measure how well they can complete their tasks as well as what problems they encounter during the test. These tests often reveal places where the users have problems understanding and utilizing the product, but also places where the users are more likely to succeed.

To be able to conduct *Usability tests* there have to be something fairly complete to test it on. This can be production software or a traditional prototype, it does not matter. What does matter is that the prototype simulate how the actual product will look, work and feel.

The results of *Usability tests* are often measureable and quantitative, and are thus suited when comparing different design variants in order to determine the most effective solution. These tests are especially good at determining:

- *Naming*: Do the naming of buttons and labels make sense? Are there certain words that may be more suitable to use than others?
- *Organization*: Are similar items grouped in categories that make sense? Are these items placed so that the customer easily can find them?
- *First-time use and discoverability*: Are items that are often used easy for new user to find? Are instructions clear? Are instructions needed?
- *Effectiveness*: Is it possible for customers to efficiently complete specific tasks? Do they make mistakes doing this? Where and how often?

It is worth noting that *Usability tests* are most suitable for first-time use of a product. It is quite difficult to determine how effective a solution is the 50th time a user uses it. Considering that products are often designed to be used by intermediates and experts this might question the effectiveness of these kinds of tests. But there are techniques to solve this, for example *diary studies* where the subjects keep diaries describing their interactions with the product. [Cooper 2007]

4.2 Heuristic Evaluation

Heuristic Evaluation is considered a discount usability engineering method, meaning that it is a cheap, quick, and easy way to evaluate user interface design. Because of this it has become the most popular of usability inspection methods. The main goal is to find usability problems in the design so that they can be attended as a part of an iterative design process. Heuristic Evaluations are done by a few numbers of evaluators that examines the interface to see if it coincides with recognized usability principles (so called *heuristics*).

It is not recommended that a heuristic evaluation is done by a single person because it is not possible for one person to find all usability problems in an interface. People are different and different people find different usability problems [Nielsen 2005]. Because of this it is possible to find most usability problem in an interface with just a few participants. Jakob Nielsen, the creator of the heuristic evaluation, recommends the use of three to five evaluators, using more people than that doesn't give much additional information.

The heuristic evaluation is done by letting the each evaluator inspect the interface alone. Not until all of the evaluators are finished are they allowed to talk to each other. This in order to make sure that each evaluation is independent and unbiased. The result can be written down in a report by each evaluator or can be verbally done to an observer that is present during each evaluation. The latter is recommended because it is easier to organize one set of personal notes than a set of reports written by others. The observer can also assist the evaluator in case of trouble or if his knowledge of the domain is limited. [Nielsen 2005]

Heuristic evaluation is not perfect; there are risks and problems using this method. The main issue is that the evaluators are not users themselves, they only tries to emulate the users. There is thus no real user feedback, unless there are actual users involved in the evaluation. Another concern is that heuristic evaluations do not scale well for complex interfaces. A small team of evaluators may miss a majority of problems and not even find more serious ones. [UPA 2005]

An important difference between heuristic evaluations and more traditional evaluations is that the observer is allowed to answer questions during the evaluation and even give hints on using the interface.

The evaluation sessions usually lasts 1-2 hours for each evaluator. Longer sessions might be needed for more complex interfaces, but if so it is recommended to split the evaluations into smaller sessions. [Nielsen 2005]

4.3 Iterative Design

Iterative design is a design methodology that is often used when developing user interfaces. It work on the premises that the developers first create a initial design or prototype of the interface that is tested, analyzed, refined and done all over again until the developers happy with the result (see Figure 4-1).

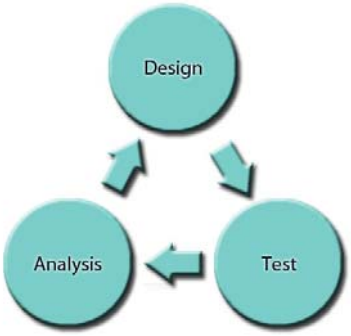


Figure 4-1: The Iterative Design Process

The reason that this methodology is especially useful for user interfaces is because it is almost impossible to design a user interface that has no usability problems (read more about *Usability* in chapter 3.1) from the start. Not even a usability expert could make a perfect user interface with his first attempt, so that is why all user interface development should use an iterative approach in order to make the interface as good as possible.

The way it works is that the developers first create an initial design or prototype of the interface. The prototype would then go through one or more tests with a number of test subjects. The problems that the users would come up against would be noted down and used to improve the user interface. This new design would then be tested again and the new problems noted would be used to improve the interface yet again. This will be done over and over again until either the developers think the interface is "good enough" or the number of problems that would be found are so small that it would be a waste of time and money to continue. How many iterations and number of test subjects that is needed differs from project to project. More complex user interfaces will probably need many iterations before most of the usability problems are found and corrected, the number of test subject need not be large, sometimes a small number is enough to get the needed information.

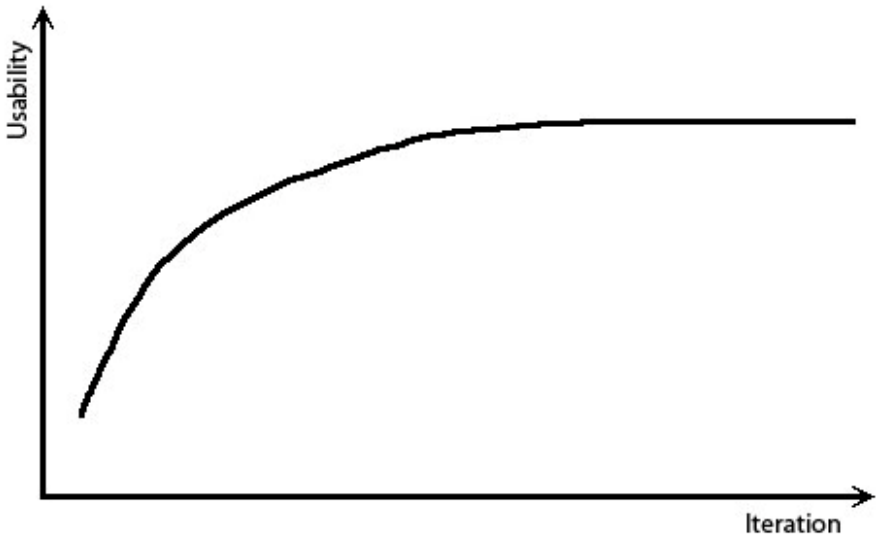


Figure 4-2: Iteration vs. Usability

Figure 4-2 above shows a conceptual graph of the relations between the number of iterations and the improvement in usability. As can be seen in the figure the curve will go upwards after each iteration early in the development, after a while though the curve will begin to smooth out and eventually it will plateau. Ideally each iteration will be better than the last one but this is not always the case, sometimes the changes made in the design will actually make the interface worse than before. The curve in Figure 4-2 should therefore be seen as an ideal example of how iterative design should work. [Neilsen 1993]

The main advantages with an iterative design process over a more classic one like the waterfall model are numerous, for example:

1. Serious misunderstandings and usability problems are discovered early in the project, when there is still time to do something about them.
2. It makes it possible for, as well as encourages, users to give feedback.
3. Finding the most serious issues early forces the development team to focus on and can thus ignore (at least temporary) other lesser issues.
4. Doing continuous iterative testing enables a more objective view of the project's status.
5. The development team workload is spread out more evenly throughout the development cycle.
6. The team will quickly get used to the iterative design process and thus continuously improving it.
7. It is easier to show the stakeholders that the development is going forward. [Kruchten 2000]

4.4 Software Prototyping

Software prototyping is a process that is used early in the development to build a model of a system, this is called a prototype. Prototypes help system designer to build informative systems, and is also a part of an iterative process (see previous chapter). Prototyping can be of use when determining the initial system requirements because it coverts these basic specifications into tangible but limited working model of the desired information system. By doing this the users will have something physical that they can touch and see and thus giving them the chance to give the developers useful feedback that can be used to modify existing requirements as well as developing new ones.

There are several kinds of prototypes, from low tech ones like sketches and traditional screens to high tech operational systems like using CASE (Computer-Aided Software Engineering) Tools and systems created with fourth generation languages, like Visual Basic [Mcclendon et al 1999]. They can be classified into three main categories, wireframes/traditional prototypes, visual prototypes and interactive prototypes.

Wireframes and traditional prototypes are good in early development to demonstrate the basic concept, although they are very limited, non-interactive and usually very broad. But its' strength lies in that it is easy and quick to create and don't require much technical expertise to do.

Visual prototypes often come in the form of screen mock-ups, done with some sort of visual editing tool like Adobe Photoshop. The main purpose of this kind of prototype is

to create the look and feel of the system, although it will be lacking any kind of real functionality and operation flow. It is actually a bit misguided to call it a true prototype; it is really just a visual mock-up. The visual prototypes are usually done from the designers' viewpoint rather than that of a business or software expert.

Interactive prototypes are more useful than the previous types; this also means that they require a more time and resources than the others to create. The purpose is to model the system design more faithfully, and make actually interaction possible, i.e. navigation, use of real web controls or even mock data processing. [Thomson 2009]

The advantages of using prototyping are many; it reduced development time and thus reduced development costs. Users are involved at an earlier stage giving the developers quantifiable user feedback to work with. Since users knows what to expect it facilitates system implementation and results in higher user satisfaction. This process also exposes the developers to future system enhancements.

Where there are advantages there are also disadvantages. Using prototypes can lead to insufficient analysis, i.e. the developers become distracted from focusing on a limited prototype instead of properly analyzing the complete system. The users may expect the performance of the complete system to be the same as the prototype. The developers may also become too attached to their prototypes refusing to dispose it or make necessary changes. There is also a risk that the use of prototyping can cause the system to be left in an unfinished state and/or be implemented before it is ready. This can also lead to incomplete documentation. Worth noting is that creating sophisticated prototypes with 4^{th} generation languages or CASE Tools may result in that the time saving benefit is lost [Mcclendon et al 1999].

4.5 Design Guidelines

Design guidelines help developers by providing useful high to low level guidance on the design of user interfaces. Sometimes the use of specific guidelines are specified as a part the usability requirements. It is important to get familiarized with the different existing guidelines as they contain years of experience in the field of interface design. It would be foolish to make mistakes that could have easily been avoided if appropriate guidelines had been used. Following design guidelines improves the quality of the interface.

There are different design guidelines that can be roughly divided into four groups. First there are general user interface guidelines that can be read in ISO 9241-10 for example. There are also guidelines for graphical user interfaces that can be found in a number of different books such as Cooper (2007) or Dix et al (2004). Thirdly there are guidelines for web pages as well, and lastly there are application-specific guidelines for special technologies. [UsabilityNet 2006]

5 Planning

The following chapter lays out the initial plan of the thesis, i.e. this is how the thesis was originally planned to be carried out. Note that this plan is not accurately describing how the thesis was actually done.

5.1 Time Plan

The original time plan can be seen in Table 5-1 below. The rest of this chapter will explain the different steps in the time plan more specifically.

	Time Plan
W18-19	Gain insight into the thesis. Meetings with people involved in the thesis. Write thesis description and time plan. Get access to computer, programs and so on. Start looking for suitable Online Help platforms.
W20-21	Go through the different platforms. Have meeting with documentation department (DP/K). Contact other people within the Saab Group about Online Help.
W22	Go through literature and create Online Help guidelines
W23	Take screenshots of software and make mock-up prototype.
W24	First mock-up done. Show supervisors. Create second mock-up from feedback. Show supervisors again.
W25	Research information about heuristic evaluations. Get access to source code for project and gain insight into how it works.
W26	Decide which platform to use and how the Online Help should be implemented. Start working on the first prototype. It should focus on implementation of Pop-up in 1-2 windows
W27	First prototype done. Show to supervisors. Do a simple usability test. Go through feedback. Try to find a tools for creating help content, like RoboHelp.
W28	Start working on second prototype. Should have a fully working online manual. More context sensitive help, i.e. F1 key help and more Pop-ups. Start working on report if not yet done so.
W29-30	Second prototype done. Show supervisors. Do a simple usability test and look through the feedback. Start working on the third prototype. This should be the final version, i.e. contain all necessary functionality.
W31	Third prototype done. Show supervisors. Have a final more structured usability test. Put together the test result.
W32	Make final changes to prototype. Write the report.
W33-36	Write report. Maybe take a week of.
W37	Report done, send to opponents. Prepare presentation.
W38	Thesis presentation. Fix report.

Table 5-1: The Time Plan

5.2 Preparation

First step was to find and read material about Online Help, support systems and similar to get familiar with the area. The second step was to talk with the people that are making the product and others that are somehow involved to see what sort of thing they expect (or not expect) from this thesis.

From this a project description would be written explaining the purpose and goal of the thesis, as well as some Human-Machine Interaction (HMI) guidelines embodying a good Online Help system.

After that it would be time to start looking for suitable tools/platforms that support Online Help in a Java/Linux environment. Each tool/platform that fulfills these criteria will be further researched and tested to see how it works out in practice. Time would also be taken to look within Saab to see if anyone else has done any work/research on Online Help.

A meeting with SMW's documentation department DP/K will also be held sometime within the first few weeks to see how the customer documentation is made, and see if there is a way to use any of this to create Online Help.

5.3 Mock-up prototype

After doing some research and getting a clear picture of how the Online Help would look and work a mock-up prototype would be made to illustrate this and later be shown to the thesis supervisor to get feedback. This feedback could then be used in a second mock-up or perhaps in the actual prototype.

In this stage a choice of what tool/platform to use to create the Online Help must be done, as well as acquire the source code to the software that the Online Help was going to be implemented in. A few days would probably be needed to study the software code in order to figure out how to implement the Online Help.

Followed by researching and deciding what type of usability test that could be used to evaluate the prototypes.

5.4 First prototype

When that was done it was time to start working on a first prototype, that would likely be very simple, just testing out the help system on the product to see how simple/hard it would be to implement and give a feeling of how it would be to use the help system.

After completing the prototype a usability test would be held in order to find the main issues with the Online Help, as well as determining what could or could not be done with the tool/platform. The feedback would then be used to create the second prototype.

5.5 Second prototype

The second prototype would focus more on the actual online manual, a part of the Online Help system. The online documentation was supposed to be created from the same material as the traditional manual, so suitable tools for doing that must be found and researched, and then used to create a simple version of the online manual. This manual would then be implemented in the second prototype, as well as more functionality and fixing the greatest issues with the first prototype.

A second usability test would then be held, maybe with the same people as last time or maybe with new ones (see chapter 4.1). This test would likely be more structured and take longer than the last one, and would probably result in more “data” to compile and analyze.

5.6 Third and final prototype

The feedback would yet again be used to create the next and final prototype that would have a "complete" online manual, and all Online Help functionality that was envisioned in the beginning of the project. This prototype would also go through a final usability test that would be even more structured than last time, more thorough and have more testers.

The feedback would then be used to make some adjustments to the prototype before "submitting" it, and later be used in the report (this report) to explain what needs to be done when creating a "real" Online Help.

The rest of the time would be used to write this report, which of course will be written alongside the rest of the thesis.

6 Making HCI Guidelines for Online Help

To get a better understanding on how an Online Help system should look and work it was decided the best course of action would be to research guidelines for Human-Computer Interaction (HCI). This was done by studying the books by Cooper (2007) and Dix (2004). These two authors seem to focus on different parts of how Online Help should be structured. In the end a summary of the most important guidelines from each book were written down, which will be described in the next section.

6.1 Design Guidelines for Online Help

Different books gives different guidelines but most of the time they do not conflict with each other. Cooper (2007) gives the following advice on Online Help:

- Online Help should never act as a crutch for the product.
- Online Help should be designed for experienced users, not beginners.
- A usable index is generated by exploring the application, not by reading help text.
- Try to think goal-directed while creating the index. Thinking like a user helps a lot. Especially since a large number of synonyms are needed create a complete and robust index as possible.
- There should be a menu item in the *Help* menu that handles shortcuts for keyboard and functions.
- There should also be a menu option that gives an overview of different functions in the application, things like scope, effect, power, upside, downside, and why the function should be used.
- ToolTips should not be underestimated, it is easier for the user to point at something he wants help with and gets a small box quickly describing it than have to open the help content that takes up a lot of space and try to look for it.
- Wizards should be avoided if possible, because they often have the tendency to interrogate the user, and also asking obscure questions. A user that does not know what an IP address is will not get any help from a wizard asking for it.
- "Intelligent" agents if used take a lot of work. Use *Clippy* (Microsoft Office little annoying assistant) as an example of how an agent should not be.

Dix (2004) also has a few suggestions about how Online Help should work. First there are some general requirements that have to be fulfilled:

- The user should always be able to access the Online Help while using the product.
- Online Help should always give correct information, i.e. if the product is updated the help system have to be updated as well to show correct information.
- Online Help should also be complete, i.e. there should not any sections of the product that is not brought up in Online Help.
- Online Help should be consistent, i.e. it is unhelpful if a command is described one way in the traditional manual, another way in the Online Help and a third way in ToolTip for example. The way the help is accessed throughout the system should not differ either.
- Online Help should be robust, i.e. the help system should still be accessible even if the product fails. It has to be more robust than the product itself.

- Online Help should be flexible, i.e. a user should get the level of help he needs. This can be done by using context-sensitive help or even fully adaptable systems that can feel the user's expertise.
- Online Help should be unobtrusive, i.e. it should not hamper the user from continuing his work or otherwise be in the way.

There are also some guidelines on how the online documentation, the help content, should be structured and written:

- The use of hyperlinks is a good way of making the documentation easy to read and navigate.
- Use a clear structure with headings that work as signposts.
- Organize information after user tasks.
- Keep the sentences short, to the point and free from unnecessary jargon. The language should be simple but not condescending.
- Do not make any assumptions of what the user already knows while writing the documentation.
- Bring up procedures in order with numbered steps. Highlight important steps.
- Use examples where appropriate.
- Support searching with the help of indexes, table of content, glossaries and free search.
- Include lists with known and common error messages.
- Include FAQ with clear answers.

Lastly there are some pointers of how the user support system should be designed:

- Online Help should not be seen as an *add-on* to the product.
- Online Help should ideally be designed together with the rest of the system.
- There are several ways the user can access the help system, commands, buttons, functions that can be turned on or off, or a separate application. Using keyboard or mouse buttons are suitable for context-sensitive help.
- The way the help system is shown is also important; an application that is run in windowed mode should show the Online Help in a separate window, while a fullscreen application should have it in splitscreen. ToolTips are useful to show short bits of information to the user.
- The documentation and looks of the Online Help should be designed the same way as the interface in the application, by taking into account the ability of the user and the user demands.

7 Creating a Mock-up

In order to get a clear picture of what kind of Online Help functionality that was going to be implemented, as well as how it would look and work, it was decided that the easiest way of doing this was to do a mock-up of the interface with the Online Help functionality implemented. This was done by taking screenshots of the current interface and then modifying them in Adobe Photoshop.

This was done before any real background check of possible Online Help system platform had been done. Because of this it was not known during this time what was possible to do with the different platforms and therefore did not influence the making of the mock-up. The mock-up was meant to show how ideally the Online Help would look and work.

The choice of Online Help functionality that was implemented into the mock-up was a *Help* menu in the menu bar and a combination of *ToolTips* and *Pop-up* windows. The *Help* menu contained four items, *Help Content*, *Keyboard Shortcuts*, *Symbol Legend*, and *About* (see Figure 6-1). The ToolTip was used show what certain abbreviations meant, explaining what kind of values that were expected in certain field and also what some of the table headers meant, while Pop-ups were used to explain certain labels, checkboxes, buttons, text fields, table headers, combo-boxes, etc. The main difference between ToolTips and Pop-ups are that ToolTips are shown when the mouse pointer hovers over certain components a short while and are usually very short (see Figure 6-2 for an example). Pop-ups on the other hand are shown when the user activate them, in this case by right-clicking on it to show a "What is this?" menu that he has to left-click on to show the Pop-up (see Figure 6-3). It closes when the user clicks on something else, and often contains a lot more text than ToolTips.

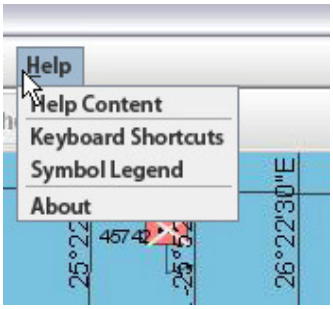


Figure 7-1: The Help menu

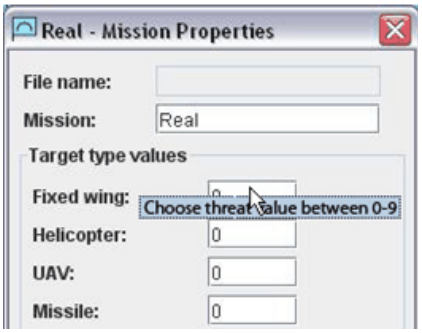


Figure 7-2: ToolTip

This mock-up were presented to the two supervisors at SMW, one programmer and one human factor engineer, that were involved with thesis in order to get feedback and useful advice that could be used to determine what exactly the Online Help should be focused on. The feedback mainly discussed if both ToolTip and Pop-up should be used, whether the use of them should be consequent or used differently from window to window depending on context. In the end it was decided that in order to be consistent the focus should be put on implementing Pop-ups on those components that are relevant since ToolTips are not suitable to show longer text masses and cannot be controlled by the user in the same Pop-ups. Other decisions were that components that had a label attached to them should both have the same Pop-up; the middle mouse button should be

used instead of the right one because it is already used to open shortcuts menus on the map in the product. No Pop-ups should be used on the group box labels (a label that defines a whole set of components), the user will have to look in the online manual for these.

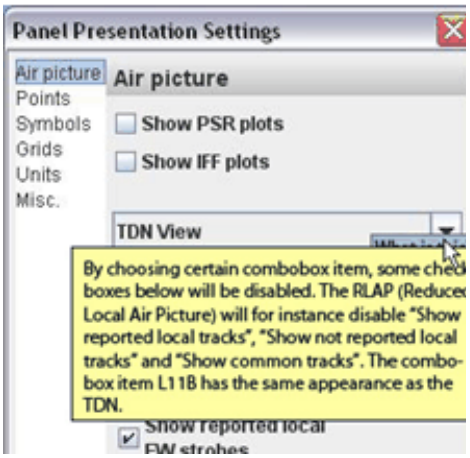


Figure 7-3: Pop-up

8 Searching for Online Help System Platforms

One of the first challenges with the thesis was to determine which platform to use for the Online Help System. Researching the subject gave the answer that there were a multitude of such platform on the market. This was done by first using Wikipedia to see what system was available; there were a few but only JavaHelp and Oracle Help were of any real interest, since they were Java based. After this some time were put on using Google search to find more examples of Online Help platforms, there were a lot of hits but most were either outdated or not Java based. The only real option that was found was Eclipse Platform Help System (EPHS). There was also some research done within Saab which resulted in two possible options, a solution using Firefox Portable and another one using ActiViewer. In order to decide which of these were most suitable each platform was tested and evaluated. The description of the platforms can be read in chapter 4.5.

8.1 Building a test GUI

In order to test the help platforms some sort of test Graphical User Interface (GUI) was needed. Because the underlying code was not important it was decided to use NetBeans to create the GUI. In NetBeans you can simply drag and drop different components on a surface and the code for the resulting GUI is auto generated. It all worked out great until the GUI was to be imported into Eclipse. The problem was that there was no way to import the project; Eclipse does not support the ability to import NetBeans projects, while the opposite is possible. Neither is it possible to copy the code into an Eclipse project because the code contains classes only available in NetBeans.

So an alternative way of doing the GUI was needed. In the end the solution was simple. The Java homepage had tutorials available for the different Swing components. Most of these had the code for the different tutorial assignment provided. The code for one of these were chosen and modified to act as a test GUI for the help platforms.

The resulting test GUI was a simple window with a menu bar, text field and a button (see Figure 8-1 below).

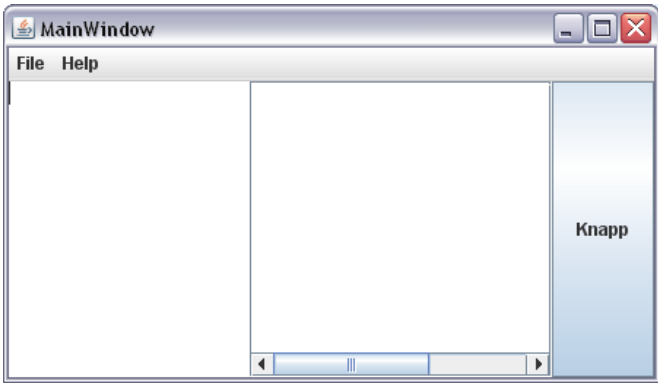


Figure 8-1: The Test GUI

8.2 Eclipse Platform Help System (EPHS)

Because Eclipse is the main programming environment at SMW it seemed like a good idea to incorporate a help system that used that environment.

8.2.1 Testing

Following the steps mentioned in chapter 4.5.1 a simple help system was created to test EPHS along with a few HTML files. This did not pose any problems; the problems began than it was time to test the help system with the Eclipse Help Content window. Eclipse Help Content window uses Internet Explorer to show the HTML files and for some reason SMW firewall blocks the viewing of these files, making it impossible to see how the help content looks or if it even works. This essentially made any further testing of EPHS pointless; working in the blind is never a good idea.

The main disadvantage with using EPHS, besides the problem of not being able to see the help content, was that in order to make the help system work, once the product had been shipped, a slimmed down version of Eclipse was needed to be installed as well. Because SWM's customers seldom let anything else than the main software be installed on their computers makes EPHS a bad choice in this case. It was then decided that EPHS was not suited for this and thus discarded.

8.3 JavaHelp

JavaHelp is another obvious and suitable choice of a help system, mainly because it is Java's own version of Online Help and therefore, in theory anyway, should work exceptionally well since Java is used in the application.

8.3.1 Testing

To test things out simple versions of the helpset, map and topic files were created, as well as some simple HTML topic files. It was then a simple thing to add a few lines of code into the test GUI. The main window as well as several components had topic IDs linked to them by using the `enableHelpKey()` method. In order to get this to work on buttons and likewise the actual button had to be pressed down before the F1 key was pressed. This seemed cumbersome in the long run so instead a small "What is this?"-menu was created that appeared when the user pressed the right menu button on a component with a topic ID. Pressing the menu item opened the help content window on the page that represented the chosen topic ID.

All in all JavaHelp was easy to use and there were no problems creating the help files. It is well documented and it is possible to configure the help content in several different ways. The downside is that not much have happened recently with JavaHelp, the latest build is from 2007, but it seems like nothing major has happened since 2004. This might be because the interest from the community has declined over the years. The question is if Sun is going to abandon any future development of JavaHelp, and if so this should JavaHelp then be used in this thesis? Another problem is that there are some things that are not clearly explained in the documentation. To find solutions to this Google must be used, but because JavaHelp does not see much use these days it is not always certain that a solution will be found.

8.4 Oracle Help

Oracle Help is Oracle's own version of JavaHelp, much of help files structure is taken from JavaHelp, but several new features have been included. The programming part has also been simplified compared to JavaHelp.

8.4.1 Testing

It was a simple matter to create the help files for Oracle Help considering that most of the help files from JavaHelp could be reused without too much modification. The same test GUI was used as with JavaHelp. The code for Oracle Help was even simpler than JavaHelp to implement and worked like charm. Strangely enough there does not seem to be any way of implementing Pop-ups in the application with Oracle Help, there is support for Pop-ups but only in the help content, i.e. it is possible to define a link that then clicked on will open the referenced page in a Pop-up inside the topic window. There is a boolean **`needPopupHelp`** that can be set to *true* in the **`addComponent()`** function, but this will only result in that the topic window will open, not the navigation window, and it need to be closed manually. So calling it Pop-up help is a bit misleading.

Considering that Pop-ups is an important part of this thesis, just making help content and show help for each window with F1 is not enough. It is important that the user also can get immediate help with components those purpose may be unclear or those meaning has been forgotten. This can easily be done with Pop-up windows, but since Oracle Help do not support these within the actual application, its uses in this thesis are limited.

8.5 ActiViewer

Before doing any testing it was decided to find more information about the product, so the distributer's homepage was visited, but no information about ActiViewer whatsoever was found. A simple Google search was also done but resulted in no relevant hits. After checking with DP/K it seems like the distributer is no longer selling ActiViewer for customers, this means that there will be no further development on it and is essentially a dead product. To adapt such a product for use as an Online Help system, that probably will be switched to something else within a foreseeable future, would be unwise and a waste of time.

8.6 Firefox Portable

A demo of one of the manuals that Saab System had made was sent to SMW so it could be tested and determine its use in this thesis.

8.6.1 Testing the demo

The demo received was a CD based version, and needed to transfer some temporary files to the hard drive to be able to run. The manual looked nice and the headings were ordered as hierarchical file tree, clicking on one of the heading would open up the topic on the same page slightly indented inside a frame (see Figure 2-5). Clicking on it again will close the frame, it all work just like the + and – signs in a hierarchical file tree. All the standard toolbars and menus had been removed and instead a toolbar containing the functions of a standard help content window had been implemented. This included

topic, index, search etc. The only thing giving away that it was Firefox was the small Firefox icon in the upper left corner.

The only problem with the manual was that it was just the manual, not the application it was meant for. It was thus no way to test the actual interaction between the application and Online Help. Neither was it possible to see how the code of the Online Help had been implemented in the application. Contacting Saab Systems about this revealed that it was unlikely that they would disclose any information about this to another Saab company in a foreseeable future; company bureaucracy at its finest. Because of this it unlikely that Firefox Portable will not be used as the Online Help system in this thesis.

8.7 Choosing Online Help platform

After looking through the available choices the answer was quite obvious. Eclipse Platform Help System did not work well within SMW's network, and needed a slimmed down version of Eclipse in order to work. Oracle Help was easier to use than JavaHelp but did not support Pop-up windows in applications. ActiViewer was a no longer supported by their developer and thus a dead end. The Firefox Portable version seemed like a good way to go, but without any API, code or otherwise way to use it there was no point of pursuing it any further. The only platform left was JavaHelp, this is by no means the most optimal platform but the only other options is either create an own version of Online Help from scratch or use JavaHelp as a base and build something from that. Both of these options would take too much time, and this thesis was never about creating an Online Help from scratch but trying to do something with what was already available.

9 Designing the Initial Prototype

After it had been decided to use JavaHelp to implement Online Help the work on the first prototype could be started. The first step was to create a new branch for the code in the source control program ClearCase, both to protect the original code and to make sure that the code for this thesis was safely backed up.

9.1 Building the Prototype

With this done it was time to start on the prototype. In order to get more familiar with the code the first thing that was implemented was a *Help* menu in the interface with the same options as in the mock-up mentioned earlier (see chapter 7).

To start things of a help class called **`MMIHelp`** was created to handle the creation of a **`HelpSet`** variable that is linked to the helpset file as well as a **`HelpBroker`** variable that is created from the **`HelpSet`** variable and handles all the JavaHelp operations. This class was written as a **`Singelton`** class so that only one instance of it can exist at a time, or else the JavaHelp would start to malfunction. To make things easy in the beginning the same help files that were used on the test GUI was used here as well. The menu item *Help Content* under the *Help* menu was then set to open the online manual when pressed.

But the online manual should also be available when pressing F1 in a window or side panel. The starting point of the Online Help implementation was at a side panel called *Weapon Engagement Status*, the reason for choosing this panel was that it contained a table as well as a few checkboxes and labels, and therefore a perfect place to start testing the *Pop-up* windows. But before that the panel was linked to a page in the online manual with the **`enableHelpKey()`** function, so that that specific page was opened when the F1 key was pressed. Implementing the Pop-up help was not that complicated either, at least not for the checkboxes. First of all the **`enableHelp()`** function was used to link topic IDs to the checkboxes and a **`mouseListener`** was added to those same component. The responding function was programmed so that when the middle mouse button was clicked on such a component a new Pop-up window would be created that showed the topic that corresponded to that component. The Pop-up window was closed and destroyed as soon as something else outside the Pop-up was clicked.

This was easily done for the checkboxes, but it was not as easy to implement in the table headers. The problem here is that the table headers in a **`JTable`** lies under a single component that can be gotten by using the function **`getTableHeader()`** on the **`JTable`**, the table headers themselves have no components of their own. The solution to this problem was to make some changes to the **`TableModel`** of the **`JTable`**. The project code contains a class called **`ColumnDefinition`** that contains information about the columns in a table, like max- and minimum size among others. This class was edited so that the topic ID was included as well for each column. In order this to actually work it has to be included into the **`TableModel`** somehow, fortunately the **`TableColumn`** class that the **`TableModel`** uses contains a variable called **`identifier`** that is not used for anything. This variable is of the type **`object`** the most basic type in Java and can therefore be set any type, in this case a topic ID. Setting it to a topic ID is done at the same time as the configuration of the table is done.

This itself is not enough to make the Pop-up window work though, the topic ID must be extracted somehow. When a **`MouseEvent`** is triggered the position of the mouse pointer is recorded, by getting the **`JTableHeader`** and the **`TableColumnModel`** it is possible to get the index of the column by using the x-coordinate of that position. The index can then be used to get the actual column and thus its topic ID. After this the same thing was done to the *Mission Control Panel*.

Instead of using the old test topics in the Pop-ups new HTML topic files were made with material from the original manual, this because the test files did not really say anything.

9.2 Testing the Prototype

With a finished prototype it was time to put it through a usability test. Three people were chosen without any special criteria to do the test. The test itself was unstructured, i.e. it did not follow any special protocol instead the testers got to play with the system for 15-20 minutes while answering a few questions. The testers were told to speak out loud any thoughts they had. The result was put together into a couple of tables, which can be seen in Appendix A. It can be summarized into positive, negative and miscellaneous parts.

9.2.1 Feedback on what was good

Interaction

The interaction was simple with a high acceptance from the test users, they especially liked to able to open Pop-up windows with just one mouse-click.

Effectiveness/Satisfaction

The quick access to help with certain components as well as the possibility to get help not only from the actual components but also from the labels next to them led to high effectiveness and satisfaction from the test users.

9.2.2 Feedback on what was bad

Window placement and management

Several of the test users did not like that they could not move or change the size of the *Pop-up* windows. There were also cases where the Pop-up was blocking certain labels which caused irritation from the testers. Some Pop-ups had scrollbars because there was too much text to be shown in its standard size, some of the testers were bothered by these others was not, but it seemed like they preferred bigger Pop-up window over scrollbars.

Interaction

One test user complained that he could not use the right mouse button to open the Pop-ups, he also did not like that the Pop-ups appeared right away. Instead there should be a small “What is this?”-like menu that is shown first, pressing this menu should then open the Pop-up.

9.2.3 Miscellaneous feedback

There were also a few suggestions of additional functionality that could be implemented to enhance usability, things like an option to have a separate help dialog

window on the side that shows help text for the components that the user is currently working on, or through the Pop-up window be able to link to a dictionary or other useful parts of the manual.

9.3 Focus of the Next Prototype

After looking over the result it was decided that the next prototype should focus on correcting the window management issue. Although it is not possible in JavaHelp to interact with the Pop-up windows the way the test users wanted to, instead the focus should be on why the testers wanted to move or resize the Pop-up windows in the first place. The reason may be because the placing of the Pop-up was handled automatically by the system and could therefore be a bit random. Fixing this should probably reduce the users desire to move the Pop-up window. To solve the second problem it would probably be easiest to create several different sizes of Pop-up windows that the scrollbars disappear. Without any scrollbars the users would unlikely want to resize the window.

10 Designing the Second Prototype

It was not only decided to fix the main issues with the first prototype in this prototype, but also to try to make a real version of the help content.

10.1 Creating the Help Content

There were two options of making the help content, one was to create everything manually with the material from the traditional manual, and the other one was to try to find some sort tool to do this automatically. As mentioned in chapter 2.4 SMW tried to make an earlier attempt with Online Help by using *RoboHelp* to make the help material. This was several years ago so RoboHelp must have improved during this time. After a quick Google search it was found out that the latest version was RoboHelp 8 and was now owned by Adobe. A trial version was downloaded to test its capabilities to create help content.

RoboHelp supports a number of different help content types; JavaHelp is only one of them. There is also an option that lets the user import FrameMaker and Word documents that will be transformed into RoboHelp's file format, XPJ among others. This seemed much easier than trying to create everything from scratch. The traditional version of the manual was in Word format, so it was imported into RoboHelp. The result was one big HTML topic file, all the headings had been converted into topic IDs, and all the pictures had been put into a separate folder and linked into the topic file. The next step was to generate the JavaHelp files to see how it would look. After configuring the JavaHelp generator the JavaHelp files were generated into a JAR file. It was then possible to view the generated JavaHelp directly in RoboHelp, doing so showed that there were serious slowdowns with the JavaHelp, loading the topic file took forever. To make sure that it was not just RoboHelp that were malfunctioning the JavaHelp files were tested in the application as well. The slowdowns were even worse, the topic page never loaded at all. It seems like the problems that Saab Systems had with JavaHelp was still there (see chapter 2.6.5).

These slowdowns are caused by the single humongous HTML topic file that the help content consisted of. Usually in a help content each topic and subtopic should have their own HTML file, so splitting the file would also make the help content seem more like the "real deal". It was decided that splitting each subtopic would be too cumbersome, so for the moment only the main topics (or chapters) would have their own HTML file. Unfortunately RoboHelp had no function that made it possible to automatically split the file, neither was there any function to do this while converting the Word document into RoboHelp format. So this had to be done manually with copy-paste. The topic IDs had to be updated so that they referenced to the new topic files. After this a new set of JavaHelp files were generated and viewed in RoboHelp. The slowdowns were completely gone. Interestingly some of the new HTML files were pretty large but this did not cause any slowdown whatsoever. It seems like the size of the files have to pass a magic border of some sort before they start to slow down the JavaHelp system.

10.2 Building the Second Prototype

The first step was to implement the new help content into the application. There was a small problem; the *Pop-ups* could no longer be used since the helpfile used for the new online manual was no longer the same as the one the Pop-ups were in. The options were to either un-JAR the help content file and manually add the Pop-up topic files, edit the map file and then make it into a JAR file again, or just make them into separate helpsets. The latter one seemed the best choice, so the `MMIHelp` class was edited to create two different helpset references, one for the online manual and one for the Pop-up help.

Next the code was changed so that F1 help used the online manual helpset and the components with Pop-up help used the Pop-up helpset. As mentioned in chapter 9.3 the issues with the positioning of the Pop-up windows have to be solved. There is a function in the `Popup` class that makes it possible to set the position of the Pop-up window. The position was set to appear a bit under the component. To solve the problem with the scrollbars in the Pop-up windows a second Pop-up window size was defined in the helpset file, which was twice as large as the original. This new size was used on Pop-ups those texts did not fit in the original size.

A strange bug showed itself while opening Pop-ups in *Mission Control* and *Weapon Engagement Status* in windowed mode. The nature of the bug is that some of the Pop-ups appear in the left corner of the window distorted instead of under the components. While this bug was mystery at first it was soon discovered that if Pop-up did not fit inside the window it was shown in the left corner instead. It seems like the Pop-up windows can only be shown inside the window they are created in. To temporary solve this problem the Pop-up window size was reduced.

Some of the test participant preferred to us the right mouse button instead together with a “What is this?”-menu. It was thus decided to implement this additional way of showing Pop-ups as a complement to the other one, giving the user options of using whichever mouse button that felt the most comfortable.

As a last touch most of the windows in the application was liked to their respective topic in the online manual, as well as creating a new menu item in the *Help* menu called *Acronyms and Abbreviations* and linked that to the similar topic in the online manual.

10.3 Testing the Second Prototype

The test was of similar structure as last time. Three new participants were asked to “play around” in the interface, mainly the *Mission Control* and *Weapon Engagement Status* panel/window. The goal was to see if the problems with the window management had been solved and if there were any new problems had appeared or not been noticed last time. The result was put together in tables following the same template as last time and can be seen in Appendix B. The result can be summarized as followed.

10.3.1 Feedback on what was good

Interaction

As the last time the test users liked the simplicity and showed high acceptance towards the use of Online Help.

Window placement and management
This time the test users hardly had anything bad to say about the window placement and management. None of the users noticed that there were two different sizes on the Pop-up windows until it was pointed to them. And one of the testers even said that he liked that the Pop-ups appeared right under the components.

10.3.2 Feedback on what was bad

Interaction
There were a lot of complaints on the interaction part this time, although not on anything serious. Two of the users did not see the point with the "What is this?"-menu, they thought it was strange that there were only one menu option, plus it was seen as an unnecessary step to have to do an extra mouse-click to open the Pop-up. This can be related to the *Iterative Design* chapter (chapter 4.3) that states it is not always that the each iteration will be an improvement. Then again this was only one of the new features implemented in this iteration. There was also some complaints that there was no [?]-button in the windows that could be used instead of pressing F1. Other than that there were just some minor things, like no help on the map or in the menus.

Efficiency
In some cases during the test the help content did not open when the F1 key was pressed, the window or panel had to be re-opened to get this to work again. The problem occurred after the test users had clicked around in the interface, it seems like the focus is moved from the window to the components and thus the F1 help no longer work.

Manual/Help Content
Because a real version of the help content was used this time it was a given that there would be a lot feedback on it. Mostly it was that the help content was not detailed enough or that the text did not make sense.

10.4 Focus of the Next Prototype

Even though much of the negative feedback was on the interaction the main issues on that part was on the "What is this?"-menu and the lack of a [?]-button. The first one can easily be fixed by not including it, and the other one is of no large importance, the interface does not need any more buttons. The focus problem has to do with the window management of the application and is already worked upon by others in SMW, although since they are working on another version of the application than the one used in this thesis their success and failure will not have any affect here. It would be to difficult and time consuming to fix this for the next prototype, so this problem will be ignored. Neither will anything be done about information in the online manual, unless a new version is made, since the making of the manual has nothing to do with this thesis.

In the end there is not much improvement to do on the next prototype, instead the focus will lie on implementing Pop-ups in the rest of the application and focus on the next usability test to make sure that the prototype can handle it.

11 Designing the Final Prototype

Even though the result from the second test did not result in any high priority issues that had to be fixed, there were still a lot of things that needed to be added in the new prototype.

11.1 Building the Final Prototype

The work was started by updating the help content; with the help of RoboHelp an index and glossary was created. The index could be created automatically with a Wizard in RoboHelp, while the glossary needed to be created manually. All the topics were checked and all references to other topics were changed into actual links.

The most tedious part was to implement *Pop-ups* in the rest of the application. As mentioned earlier each unique Pop-up has a unique topic HTML file, the files were saved in hierarchical way. All in all there were around 180 topic HTML files and 30 folders. Creating all these HTML files were not that difficult, just tiresome. Something that was a bit difficult was implementing Pop-ups in the table of the *Air Track List* panel. That table was custom made by SMW and did not work the same way as the JTable used in the rest of the application. It took a while to figure out how to link the topic IDs to the column headers and the problem was solved

Around this time the bug that made the Pop-ups appear in the left corner started to make itself known again. This happened in the *Air Track List* in window mode as well, this window has a table with over 30 columns, and the window therefore has a scrollbar. The bug happens when Pop-ups are shown in the columns that are normally hidden by the scrollbar. This seemed strange at first but it soon made sense, because the Pop-ups in the tables are set to open right under the first column, this works fine on those tables without scrollbar. The problem is that when the first column disappear out of sight while scrolling right, the Pop-up can no longer be shown at those coordinates and therefore appear in the left corner of the window. Fixing this was easy; the Pop-up location was just set to the x-coordinate of the mouse pointer. Doing this also makes the interaction seem more intuitive when the Pop-up window appears right where the user clicks on the middle mouse button. After realizing this all Pop-up locations were set to show at the x-coordinate of the mouse pointer.

This was not the only place that the bug happened. Most of the windows in the application were pretty slim and thus made it hard for the Pop-ups to fit under some of the components, also this problem only became worse when the Pop-ups were set to open at the mouse pointer. To fix this problem a number of things were done. Some windows were made wider, other windows that had very few components had Pop-ups that said that the user should check the help content instead, but mostly the locations of the Pop-ups were restricted to certain positions, for example if a certain Pop-up window's location passed a certain coordinate its position became fixed.

11.1.1 Adapting the prototype to the next usability test

The following usability test would let the test users do a task where he or she would have to create a filter in the *Tactical Data Net (TDN) Control Panel* window. To make this test as successful as possible extra time and effort were put into the *TDN Control Panel* to make the Online Help as helpful as possible. The help content topic regarding

this window and its sub-windows was rewritten so that the control and operation chapters were integrated together, in the traditional version these are two separate manuals. Extra care was also made to make sure that there was Pop-ups on all important components.

Before the test the whole prototype were looked over, it was then something strange was noticed. A number of windows that had F1 context sensitive help were no longer working, i.e. pushing F1 resulted in an *exception* from the application, it did not crash, just sending error messages whenever F1 was pressed. Those windows had worked perfectly just a few days earlier and no change had been done in them since, so the origin of this *exception* was a mystery, and still is, the only conclusion that have been drawn is that it has something to do with the helpset, but what is unclear. Fortunately this did not happen to the *TDN Control Panel* window and the usability test was thus unaffected.

11.2 The Third Usability Test

As mentioned in the previous chapter the third usability test would let the users do an assignment. The assignment was given in paper form that the testers had to read before the test, and is shown in Appendix C. It focuses on making a *TDN* filter that removes airplanes that are landing and taking of from a fictional airport on the tactical display.

Six people with varying backgrounds with no earlier experience with this system were chosen for this test. Half of them would use the traditional manual (the control and operation manual) to help them through the assignment and the other half the Online Help. The purpose was to see if the Online Help made it easier to complete the assignment and which of these two help types the participants preferred to use. The hope was that the test would show that Online Help was the superior choice, and could be used to convince SMW's customers that Online Help was the way to go.

During the test the participants were observed and notes were taken of their progress, seeing how many parts of the assignment they completed, how they used the manuals and Online Help, questions and comments, among other things. The observation protocol used for the test can be read in Appendix D.

When the participants felt like they had completed the assignment, "felt like" because unfortunately it was not possible to see the result of the filter unless the application was connected to an actual C2 Unit. The participants were questioned about how they solved the assignment, what they were thinking when solving specific parts of the assignment especially if they had failed some part. Afterwards the test was repeated with the other manual type, i.e. the Online Help users got to use the traditional manuals, vice versa. This to see if they completed the assignment differently and how they used the different manual type. Note that by this point they already knew how the assignment was supposed to be solved, this was only done to get a spontaneous picture of what they though about the other variant of the manual.

After the test had been completed the test subjects had to fill out a questionnaire with some questions about the manual type they had used. There were also possible to give suggestions and other comments. The questionnaire can be found under Appendix E.

11.3 The Test Result

The result from the third usability test showed clear differences between the test subjects using the traditional manuals and those using the Online Help. The result is summarized in Table 11-1 below. It shows that it took twice as long to complete the assignment for those that used the traditional manuals compared to the Online Help users. Note that these are mean values; the individual results differed from each other, which can be seen by looking at the standard deviation. The number of questions asked by the first group during the test was three times as many as the second group, but there were no large differences between the numbers of subtasks completed by the two groups.

		Traditional manual	Online Help
Completion Time:	**Mean value:**	17	8.5
	Stan. dev:	4.582576	6.363961
	Questions:	6	2
	Comp. tasks:	**19**/21	**17**/21
User type:	**Trial-&-error:**	1.5*	1
	Read all help:	1	1
	Hybrid:	0.5*	1
	Preferred help:	0	6
Question 1:	**Mean value:**	5	8
	Stan. dev:	2.645751	1
Question 2:	**Mean value:**	5	8
	Stan. dev:	1	1
Question 3:	**Mean value:**	5	7.333333
	Stan. dev:	1	1.154701
Question 4:	**Mean value:**	4	9
	Stan. dev:	0	0
Question 5:	**Mean value:**	3.666667	7.333333
	Stan. dev:	1.527525	1.527525
Question 6:	**Mean value:**	5	8
	Stan. dev:	1	1

* One user switch from being in one category to another mid-test

Table 11-1: Summery of the test result

During the test the way the testers were using the help material were observed, they were then placed under one of three categories:

- Those that use trial-and-error until they run into problems and then read.
- Those that read short excerpts and then solve a small part of the assignment and then read again and so on.
- Those that read large portion of the text before starting to do the assignment and then try to solve as much as possible before reading again.

Observation done during the test showed that there was almost one person in each category in the Online Help group and similar in the traditional manual group except that one participant was part of two categories at the same time, i.e. he started as one type and then changed his way of using the manual in the second half of the test.

The answers from the questionnaires (Q 1-6) showed low to medium values from the people using the traditional manuals and high to very high values from those using the Online Help.

More details about the result can be viewed in Appendix F.

12 Analysis of the Third Test Result

The result from the third usability test (see chapter 11.3) can be interpreted in different ways. For one there is the number of questions that the participants asked during the test. Just looking at the statistics says that the ones using the traditional manuals asked three times as many questions, but when looking at what they actual asked it was simple questions like where the airport in the test was located. These types of questions would probably been asked regardless of help type, so in the end these result have no weight or meaning.

Those that used the Online Help to complete the assignment did it faster than those that used the traditional manuals; on average it took twice as long for the traditional manual users to complete the assignment. However the standard deviation was much higher for those using the Online Help. One of the reasons for this was that one of the test subjects completed the assignment in four minutes in a trial-and-error fashion (read more about this later in this chapter and see Appendix D) almost without using any Online Help whatsoever. It is reasonable to believe that if that participant had been a part of the other test group the outcome would have been the same there as well. The result is therefore not entirely accurate, but still the two other test subjects in the group completed the assignment faster than the ones using the traditional manuals so the results in this test still holds merit. The result would probably been clearer if more people had been used in the test.

The number of completed subtasks differentiated only a little from each other, those that used the Online Help completed a few more subtasks than those that did not. Note that 17/21 in Table 11-1 means that the group completed 17 out of 21 subtasks, i.e. 7 subtasks per person. This could mean that the interface were designed in such a way that it did not matter what help type was used, and thus it does not matter how good an Online Help is, it does not excuse a poorly designed interface.

It was noted during the test that the way that the users used the help differed, as can be seen in the last row of the observation protocol in Appendix D. Some people, mostly men, used trial-and-error, refusing to use any help until they got stuck. Then there were those that were completely different, reading large sections of the text, trying to get an understanding of the interface before starting to solve the assignment. In between were those that read a short section and solved a small part of the assignment then read another section and solved another small part and so on. This shows that each person uses the help available differently. There are those that will not use the help available no matter how good or easy to use it is unless there is no other way. Worth noting is that when they got to try the other help type some of them used it differently than the first type. But it is hard to draw any conclusions from this since they had a so called learning bias from the first attempt so it is possible they just looked through it without actually read anything in particular.

The background of the participants seemed to matter in the way that they completed the assignment. Those that worked with testing and verification of systems had an easier time completing the assignment since their occupation means that they likely have a better knowledge of how the design philosophy for this kind of system works, i.e. their mental model of how the assignment should be solved is better compared to the others. A trained operator thus solves the assignment faster through learning than an untrained

one. This does not necessary mean that the system is intuitive and easy to understand, it is just that these people have learned how the system works and have an easier time to understand them while those that have not has to use the manual or help system much more.

In the questionnaire the test subjects had to fill out (see Appendix E) there were a question about which manual type they preferred to use. All of them answered Online Help, this can mean that all of them actually preferred it over the traditional manual in this test, or that they simply preferred Online Help in general, not necessary the one in the test.

The answers to the questions in the questionnaire can mostly be tied together to the amount of time it took to do the assignment. Especially question 1, “How long did it take to find the right information?”, plays a big role in how long time it took to complete the assignment. The other question plays a more or lesser role in this as well; question 4 for example asks how easy it was to learn how to use the help, or question 3, “To what level did you experience that you got the help needed to reach your goal?”. The rest of the questions also affect the completion time of the assignment and thus it is easy to see why the Online Help group got better completion time than the other group.

13 The Function and Appearance of the Prototype

This chapter will describe how the final prototype looks and work, but also talk about the concept of this Online Help system, i.e. how it would work ideally. The first part will describe this concept, the second will handle the help content and thirdly the context-sensitive help in the application.

13.1 The Concept of this Online Help system

Ideally this Online Help system would have consisted of a complete online manual (help content) and fully integrated context-sensitive help. The help content would be a combined version of the operations manual and the controls manual. Each chapter and following subchapter would have its own topic page (HTML file) with its own link in the Table of Content. It would contain a complete index with a lot of synonyms to make sure that users find the topics they are looking for. There would also be a glossary with explanations for acronyms and abbreviations, and clicking on an abbreviation in the text would transport the user to the glossary tab for an explanation. The help content would also have a search function that the user could use to search for certain words, possibly instead of using the index. It would also be possible to bookmark topic pages for quick access or just to remember important topics.

The context-sensitive help would be divided into two separate parts; the first would be the possibility to access topic pages in the help content for the window or sidebar that the user is currently using by pressing the F1 key. This would work for any window or sidebar no matter what size or number of components in it. The other part would be the possibility to middle click on any component (maybe not those that are self explanatory), i.e. text field, drop-down box, checkboxes, radio buttons, buttons, labels, etc., to show a Pop-up window shortly explaining what the component is and what it does. The size of the Pop-up window adapts automatically after the amount of text in it, although since a Pop-up is only meant to contain short summaries the size of the Pop-up windows will be kept reasonable. The style of the Pop-ups would closely resemble the rest of the interface in order to not look out of place.

13.2 The Help Content

The help content window is divided into three sections, a toolbar, a navigation panel with several tabs, and a topic window, all of which are shown in Figure 13-1 below.

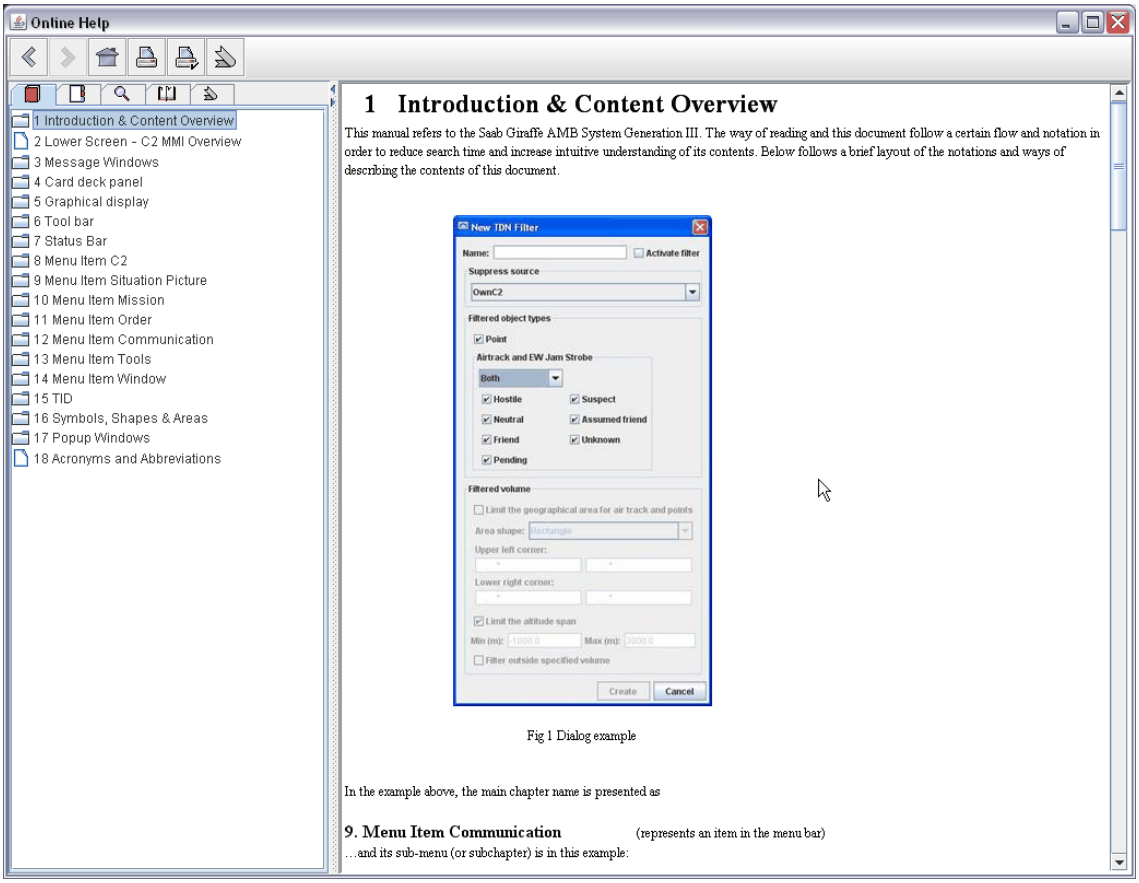


Figure 13-1: The Help Content window from the prototype

13.2.1 The Toolbar

The toolbar contains six button icons, that can be seen more clearly in Figure 13-2. Butcon one and two handles navigation back and forward, and the third is the home butcon, like a web browser. The fourth is *Print*, next is *Page Setup* and the last one is *Add to favorites* which puts a bookmark of the current topic in the favorites tab.

Figure 13-2: The Toolbar

13.2.2 The Navigation Panel

This panel contains five tabs, the first is the Table of Content (TOC) tab followed by Index, Search, Glossary and Favorites tab.

Table of Content tab

The TOC tab as the name says contains the table of content of the online manual. It has a hierarchical structure just like a file tree in windows explorer (see Figure 13-3). Each item is mapped to a topic file, all the items on the same branch is linked to the same topic file, but to different places.

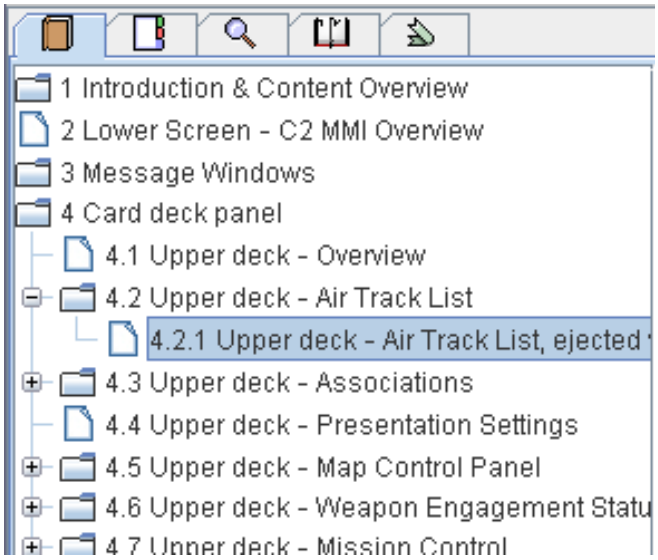


Figure 13-3: The Table of Content tab

Index tab

An index contains a list of key items in alphabetical order, like *Available IFF* followed by *Base Defence Zone* and so on (see Figure 13-4). It works just like an index in a book, except when a key word is clicked the user will immediately be moved to the topic containing that key word.

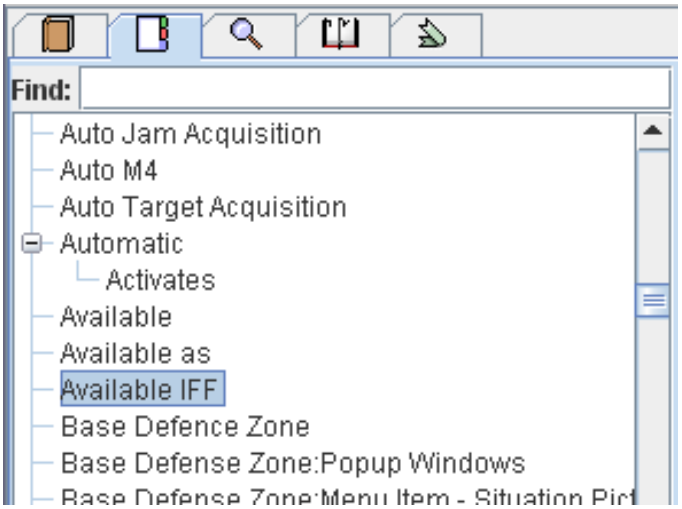


Figure 13-4: The Index tab

Search tab

In the search tab the user can search the online manual to find whatever he or she is looking for; the word *filter* for example would give the result shown in Figure 13-5. Clicking on one of the resulting items will open the topic that that word is written on that specific place with the word highlighted.

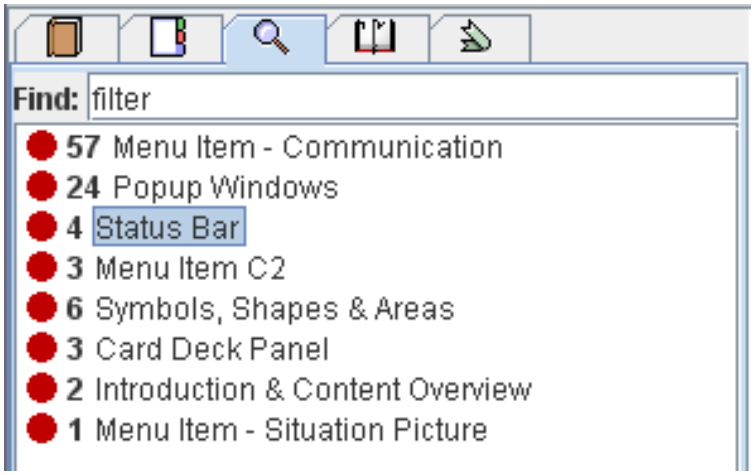


Figure 13-5: The Search tab

Glossary tab

Acronyms and abbreviations are shortly explained in the glossary tab (see Figure 13-6). There were supposed to be functionality that made it possible to connect these abbreviations in the topics to the glossary, so if the user clicked on an abbreviation the glossary would automatically open and show that specific abbreviation, but this was never implemented.

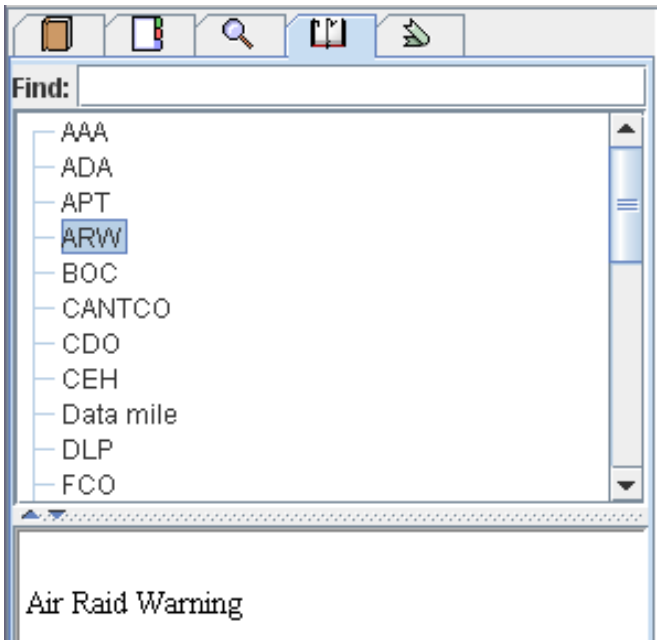


Figure 13-6: The Glossary tab

Favorites tab

The tab will be empty in the beginning, but when the user clicks on the *Add Favorites* butcon in the toolbar a bookmark will be added for the currently opened topic in the favorites tab (see Figure 13-7). Clicking on it later on will immediately open that topic in the topic window.

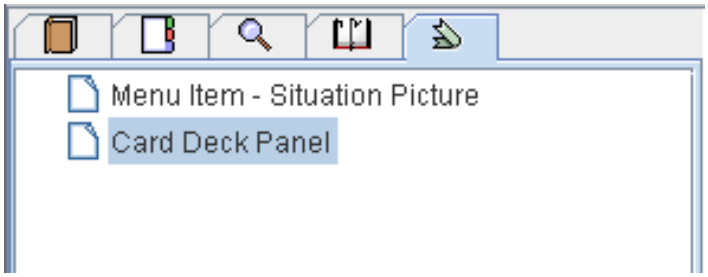


Figure 13-7

13.2.3 The Topic Window

The topic window shows the HTML topic files that are linked to the TOC and Index (see the right portion of Figure 13-1).

13.3 Online Help in the Prototype

The main part of the Online Help in the prototype apart from the help content is the context-sensitive help. Context-sensitive help is, as the name states, help that is given differently depending on the context. There are two main context-sensitive help types in the prototype, F1 key help support that opens the help content on the particular topic that handles that window/panel. The other one is Pop-up help that are shown for a number of components in the interface, buttons, text fields, checkboxes, etc.

Aside from that there is also another minor but important part of the Online Help that have not been mentioned yet and that is the *Help* menu.

13.3.1 The Help Menu

The help menu, as in most application, is the last menu in the application. It has six menu items, *Help Content, Acronyms and Abbreviations, FAQ, Keyboard Shortcuts, Symbol Overview* and *About* (see Figure 13-8).

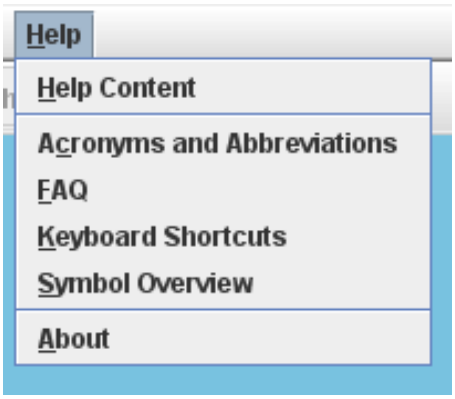


Figure 13-8: The Help Menu

The *Help Content* menu item opens the online manual mentioned in chapter 13.1. *Acronyms and Abbreviations* will open the help content with the topic about abbreviations, has the same information as the glossary tab mentioned in chapter 13.2.2. *Keyboard Shortcuts* and *FAQ* have not been implemented, but their name clearly states what they are supposed to do. *Symbol Overview* also opens the help content at the appropriate topic. Last there is the *About* item that are supposed to open a

window showing system and company information, but alas this has not been implemented either.

13.3.2 The Context-Sensitive Help in the Prototype

The context-sensitive help works the same way in the whole prototype but looks a bit different depending on where it is used.

The side panel

The side panel is placed at the right side of the interface and contains a number of different panels that can be reached by using a dropdown menu (see Figure 13-9). Going through them all would take too much time and would be completely unnecessary since most of them work and look the same. Instead the focus will be put on the first panel in the list the *Air Track List* panel. This panel contains a lot of information about the air traffic, which is shown in a very large table. If the F1 key is pressed the help content will open at the topic explaining this panel (see Figure 13-10). Clicking with the middle mouse button on one of the table column headers will show a Pop-up window placed under and to the left of the column header explaining that particular column (see Figure 13-11). As mentioned earlier the table is very large, too large to be studied in the panel. Because of this it is possible to eject the panel onto the screen in a windowed mode. This is done by clicking the eject button in the upper left corner of the panel (see Figure 13-12).

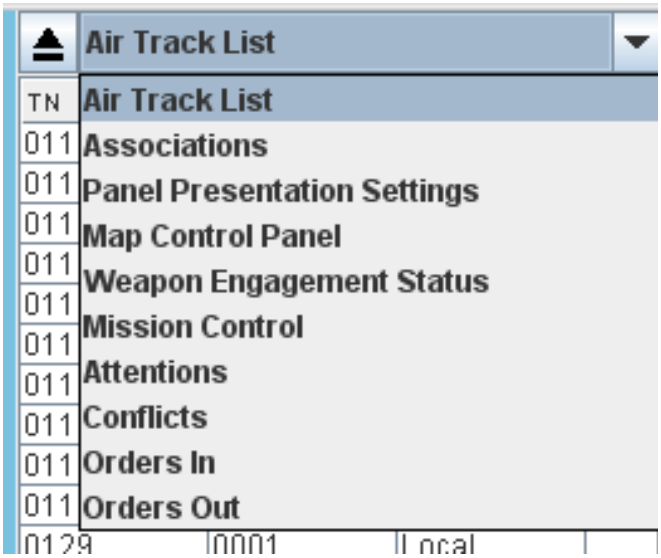


Figure 13-9: Dropdown Menu

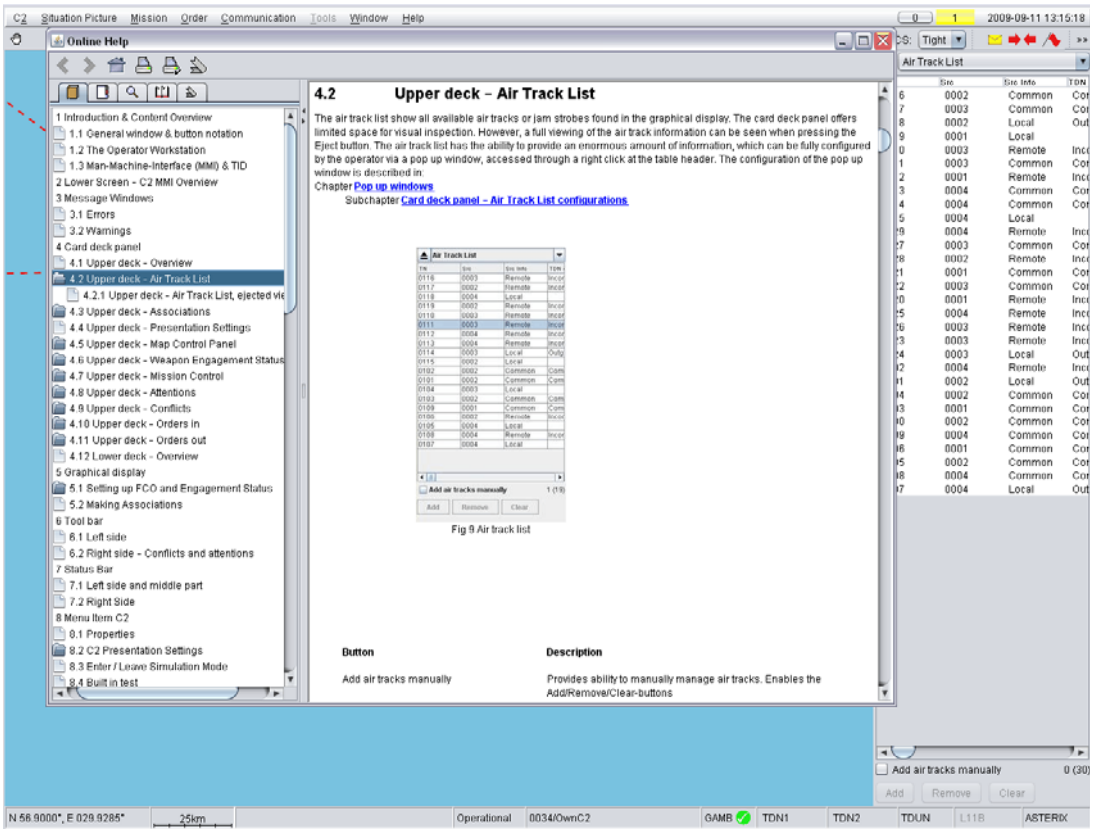


Figure 13-10: Open the relevant help topic with F1

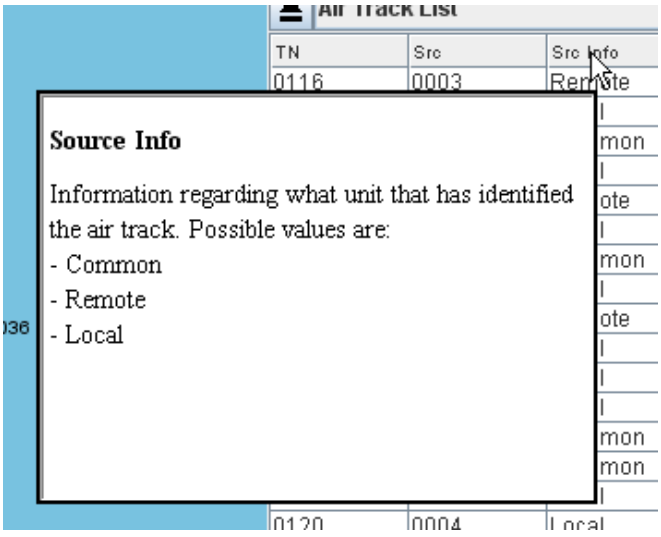


Figure 13-11: Pop-up window showing help

Figure 13-12: Eject button

Once ejected the window can be moved and resized as any other window. The table can now be studied in more detail. There are around thirty columns in the table and it thus necessary to use the scrollbar to see it all. The Pop-ups works the same way as before, except that it is shown under to the right of the column header instead of to the left of it (see Figure 13-13).

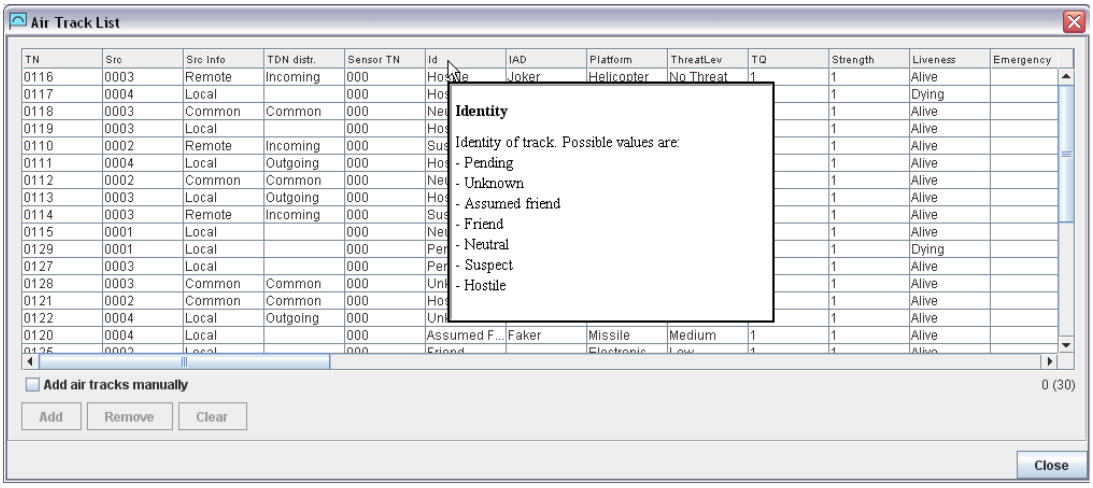


Figure 13-13: Pop-up help in ejected mode

The menu item windows

The other part the context-sensitive help are the windows which are accessed from the menus. The context-sensitive help in these windows works the same way as the ejected windows previous mentioned. The only difference between them is that there are normally a lot of more components and therefore more Pop-ups. The *TDN Control Panel* that was used in the last usability test contains a lot of Pop-ups; see Figure 13-14 for several examples. Although far from all the components have Pop-up help because most of them are self explaining enough to not need them.

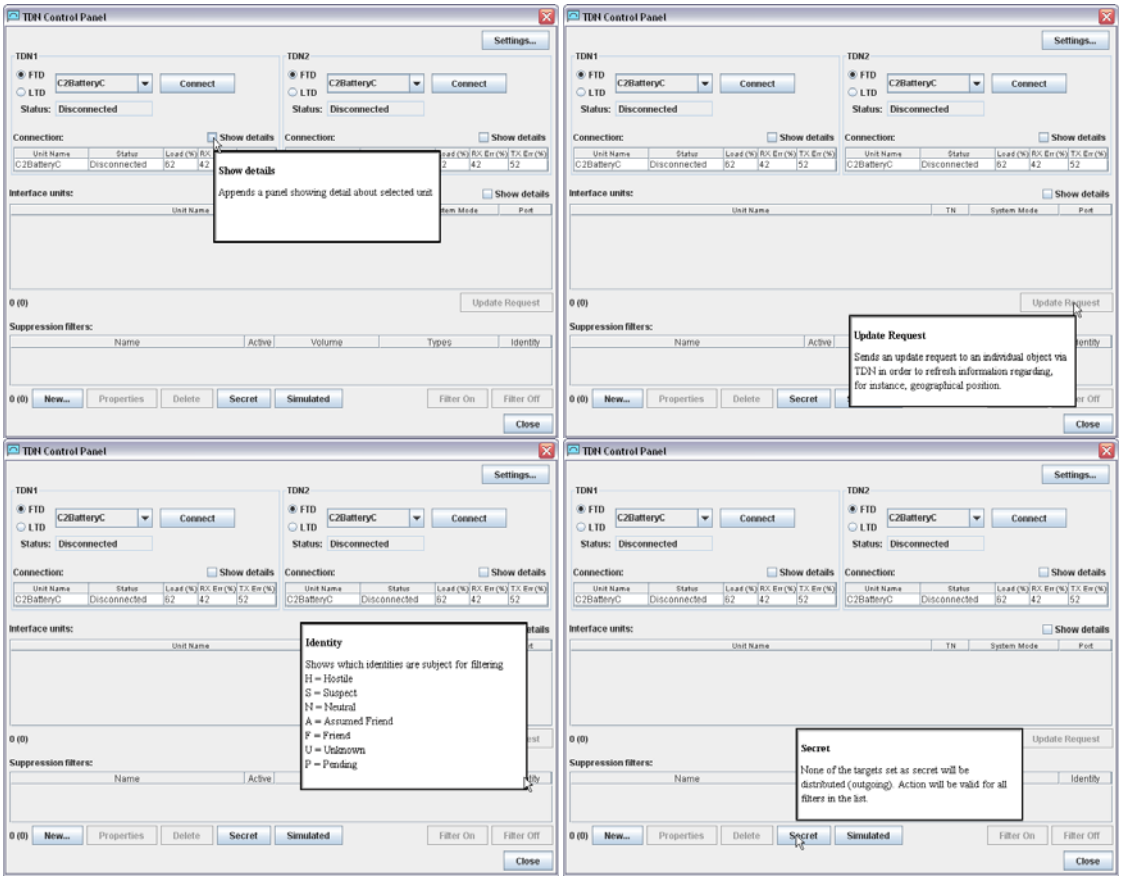


Figure 13-14: Pop-ups in the TDN Control Panel window

There are also windows within windows that have context-sensitive help, clicking *New…* in the *TDN Control Panel* window will open up the *New TDN Filter* window. That window has F1 key and Pop-up help as well. It is also possible to open the Pop-up windows by using the right mouse button instead of the middle one. The difference is that a menu with a single menu item with the text "What is this?" will be shown first, by clicking on the menu item the Pop-up will be shown (see Figure 13-15). This functionality is only implemented in *TDN Control Panel* and its underlying windows, as well as the *Weapon Engagement Status* and *Mission Control* panel, all for testing purposes.

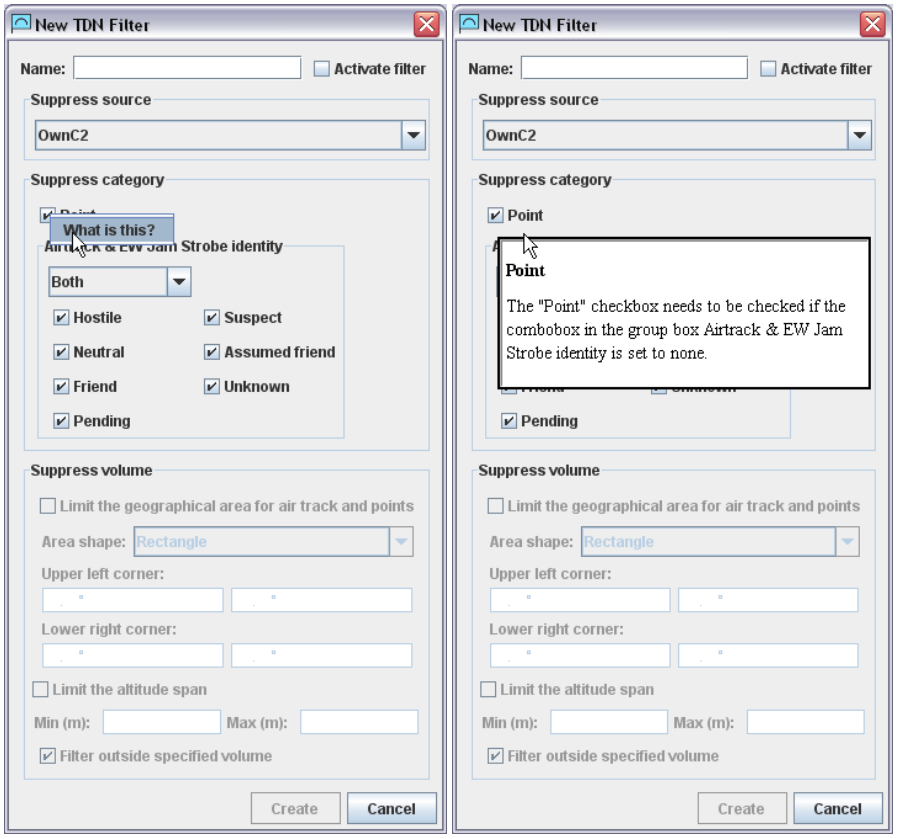


Figure 13-15: Pop-ups

14 Discussion

The main purpose of this thesis was, as mentioned in the Introduction, to:

> *Implement a prototype Online Help system in an already existing Saab product and determine the possibility of using already existing documentation to generate the content of the Online Help.*

This purpose has been fulfilled and will be brought up in more detail in the rest of the chapter. There were also few other tasks, mentioned in the introduction, which needed to be addressed before the actual prototypes could be built. These will be discussed here.

How can it be made useful *and* usable*?*
Chapter 3 explains how it should be done, and if the Online Help prototype achieved this or not is explained in chapter 14.3 below.

What is expected from the Online Help?
It was expected that the Online Help would be a good replacement for the user manuals and tests made during the thesis would give information SMW needed to convince the customers of this as well.

Determine what runtime platform to use.
It was decided to use JavaHelp to implement the Online Help. JavaHelp was chosen because it had the functionality needed to implement the Online Help illustrated in the mock-up. One can thus ask if this was the best platform for SMW to use. This is not certain but there has been a great interest in implementing Pop-ups in future projects and the only platform found that supported this was JavaHelp.

Find a program that makes it possible to generate content for the Online Help using existing documentation.
A suitable program was found to do this called RoboHelp, an Adobe product, using this it is possible to import Word or FrameMaker documents and transform these into a RoboHelp's file structure, that can be used to create a number of different help types, among them JavaHelp. This program is far from perfect considering that it only create one big HTML topic file from the imported document (at least Word documents). This topic file must be manually divided into several smaller topic files to work and look correct. A lot other small modifications also have to be done, but still using RoboHelp is a far better solution than doing it from scratch. Worth mentioning is that it was never tested to import a FrameMaker document because the version of FrameMaker installed on the workstation was too old and there was problem downloading a trial version of the latest version from Adobe for some reason. Importing a FrameMaker document may give a better result because both FrameMaker and RoboHelp are Adobe products.

The rest of the chapter will discuss the prototype, the development process, if this thesis will be useful for SMW, problems that made the authoring of this report problematic, the test results from the last usability test, and future work on the prototype.

14.1 The Finished Prototype

Of course the prototype is far from perfect; it is a prototype after all. The main problem is the use of JavaHelp to implement the Online Help. JavaHelp does not seem to be a fully developed product yet, and probably never will be since there have been no major changes in it since 2004. The bugs and shortcomings are many; window management is one of the greater ones. For example, the Pop-up windows are created inside the window which they are shown, this has the unfortunate side effect that the Pop-up windows cannot be appear outside that window. This is not a problem as long as the window is of considerable size, but in small windows there may be problems to show Pop-ups without either of them covering the whole window or making them so small that scrollbars would be needed to show all the text. Understandably neither of these solutions are good ones so it might be better to not use Pop-ups at all in small windows.

Another window management problems is that if a Pop-up is open in a window when someone closes that window using the [X]-button in the right upper corner, the Pop-up does not close, instead it will just become invisible. If the same window is opened again the Pop-up will become visible again and there is no way of closing it without having to restart the whole application. This bug does not happen when the window is closed using the OK- or Cancel-button in the lower right corner.

A third window management problem is that the Pop-up's size has to be defined beforehand in the helpset file, resulting in a number of Pop-up templates with different sizes so that different text masses can fit. This makes it difficult writing the code because the text each topic ID must be checked first to see which Pop-up size is needed. It would be better if the Pop-up windows could automatically adjust the size of the window to fit the text.

There were also problems with making the help content window open when the F1 button was pushed. Apparently JavaHelp have trouble with what part of the interface that is currently having focus. It all works fine if F1 is pressed as soon as a window with Online Help implemented is opened, but after using that window for awhile the focus is shifted from the actual window to the components and the F1 key does no longer work. Considering that the users are more likely to press F1 after trying to figure out the window, rather than pushing it right away, can cause problems.

Other problems indirectly caused by using JavaHelp are the lack of support available. There are several forums where questions can be posted but there does not seem to be much activity on those anymore. This makes it difficult when problems occurred because there is often nowhere to turn.

So the big question is, why use JavaHelp if there are so many problems using it? The simple answer is because there no other suitable platform available, the only other options is to create a new Online Help system from scratch and that would take too long.

Not all problems can be blamed on JavaHelp; some parts of the interface can be a bit confusing and need to be redesigned. Then again the application used to implement the Online Help was not done yet since the interface is still under development.

14.2 The Development Process

The thesis did not completely follow the original development plan mentioned in chapter 5. First of all the mock-up was made before any real research was done on the different Online Help platforms, so there were no knowledge if it was possible to do the Online Help illustrated in the mock-up with any of the available Online Help platforms. No second mock-up was made because the first one gave a good enough idea of what kind of functionality the Online Help needed to have. But if none of the Online Help platforms had supported the functionality in the mock-up a second one would have been made with the functionality actually available in mind.

After this the plan was more or less followed with a few minor deviations. It was just the end phase that differed a bit. The outcome from the third usability test was supposed to be used to make some final adjustment to the third prototype to finalize it. This was never done; instead the test result was used to determine how good the Online Help was vs. the traditional manual as well as determine what needed to be done in the future (see chapter 14.6).

Things also took longer to do than expected, one of these things were the last usability test. Since the two first tests were of a pretty unstructured nature there was no need for extensive planning, neither were there a problem to find people to do them. The third usability test were a bit different, in this case the test had to be planned more carefully since the test subjects would have to complete an assignment given to them. This assignment, to create a filter for air traffic, needed preparation, the assignment needed to be formulated in an understandable way, a questionnaire and observation protocol needed to created as well. Manuals needed to be printed, a room to carry out the test needed to be booked and rigged with all the necessary equipment (the earlier tests had been done at the workstation), not to mention getting at least six people with varied background and experience to do the test on a short notice. Considering that the invitation went out on the last official vacation week to ten people, and the test were supposed to be held the beginning of the following week, it was a really fortunate that there were six that agreed to take part of the test. All this resulted in that the third usability test was held two weeks later than originally planned.

There was also plans of using two versions of the application in the test, one with Online Help and one without. But it was not until the day before it was realized that this was not possible. At the time of the fist prototype a “copy” of the current version of the application was made so that it could be used to implement the Online Help. The development of the application did not stop because of this of course, so at the time of the third usability test it had changed so much so that it was no longer suitable to use in the test. Neither was there an unchanged copy of the version with the Online Help that could be have been used instead. Since SMW used a version control system that backs everything up, it was in theory possible to go back and recreate a copy of the version used in the Online Help. But this would take too much time and effort to do however, so in the end the version with the Online Help was used by both groups in the test, even those that used the traditional manual.

Could this had been planned better, probably, but considering that the thesis supervisors at Chalmers and SMW had their vacation during this time and since their expertise were needed to do the planning and execution of the last usability test got delayed.

The most time consuming part of the thesis was to implement all the Pop-ups in the interface, not only in the code but actually create the HTML content used for each Pop-up. Most of these Pop-ups will probably never be used, but since it is hard to know which Pop-ups that were going to be used in the final test, and is going to be used in the future, it was better to be safe than sorry. SMW could learn from this and make sure that Pop-ups are only implemented where they are really needed. If the implementation of Pop-ups is going to be used in many future projects it may be a good idea to develop some sort of tool that could help with the creation of the HTML content used for the Pop-ups.

During the development there have been some discussions with the Chalmers' supervisor about the use of the term 'Online Help'. Since it contains the word 'Online' one may think that it has to do with internet somehow, but this is not entirely right. The way this thesis implements Online Help does not have anything to do with internet, since all the content is located locally on the workstation. But there are other applications with Online Help that actual are online, they usually store their help content on some company server that the application has to connect to before showing the help. There are pros and cons with doing it this way, the pros are that the company can easily update the information if needed without having all their consumers download updates, instead they can just change the version of the help they have on their server. The downside is that if the customers lose their internet connection for some reason they can no longer use the help content.

The Online Help in this thesis could have been made online but there were no real reason for doing it, there is just one version of the help content, plus that the operators that are going to use it usually sits in a C2 unit somewhere in the middle of nowhere without any internet connection, so keeping it locally together with the prototype made sense. Changing the name from Online Help to something else because of this does not make sense since Online Help is what these sort of systems are called by people, no matter if it is online or not. Changing the name would just bring confusion.

Because this is a thesis done at a company that produce products for military use there are certain parts of the thesis that is classified for people outside SMW. This would in most cases been a problem but fortunately the application that the Online Help was going to be implemented in was not entirely classified. How the interface looked and worked was unclassified but the actual code and specs of the system were, as well as whom the system was meant for. Since the supervisor at Chalmers did not need to see the code or know the specs, just see the interface this did not pose such a great issue, except that he could not test the prototypes himself since they were not allowed to leave the building. Technically he could have come to SMW and test it here but since he could look at screenshots of the prototypes and get a first hand explanation of the how it worked he thought that it were enough.

All this also ties into the fact that the end users could not be used to test the prototype, since they are military personal from another country. Instead people within SMW were used to test the prototype since they had the necessary experience to do this. Also since this thesis is not meant for commercial use as well as having a limited budget, this seemed like a feasible solution.

The supervision of this thesis has been a bit different. At SMW one of the senior programmer were appointed as supervisor for this thesis. He have been a great help with everything programming and Eclipse related, but the one that have put down most time helping out with this thesis is the Human Factor Engineer (HFE) that work with interaction design and usability at SMW. Considering his background and the focus of this thesis his expertise has been most helpful in the development of the prototypes and usability tests. He sort of became the unofficial supervisor of this thesis. Having two supervisors had its pros and cons; it was easier to get help with different aspects of the thesis, both in the area of interaction design and in programming, but mostly interaction design. This led to that the official supervisor supervised less and less of the thesis and thus was not fully aware of what was going on. In the end this did not really matter since the unofficial supervisor knew what was going on.

The contact with the supervisor from Chalmers was mostly to make sure that the thesis went in the right direction and give guidance on what to do. Since some of the things in the thesis were classified he could not give any detailed guidance on the prototype or coding. Most of the tutoring was on how this report should be written.

14.3 Usefulness

One of the major results from this thesis is that a new project is in planning. This project will look into the possibility to use context-sensitive help, mainly Pop-ups in similar systems like the one used in this thesis. This may sound like a repetition of this thesis, but it is more focused on using Pop-ups in tables to explain what the different column headers means. Today meetings are held regularly discussing what the text in the column headers should say, if an abbreviation should be used and if the users would understand it. If a Pop-up window was just a mouse click away this would simplify things, an abbreviation could easily be used if a user could find out its meaning just by clicking on it and read the Pop-up. There are also long-term plans of introducing online manuals (help content) into their systems, but this would take too many resources out of their current project budget, so for now they take one step at a time.

Another important thing is how well the result from this thesis can be used in other systems. JavaHelp can easily be integrated into any system written in Java, but JavaHelp as it is right now is not recommended for commercial use. The code form the prototype should instead be used, if it should be used at all, to show how JavaHelp can be implemented in an existing system and use it to show how Online Help would look and work in that particular system.

14.4 Writing the Report

The main difficulty with writing this report was that it was done alone which means that there was no partner to discuss it with as well as someone to read the written text; fortunately the supervisors here at SMW filled this role instead. The supervisors could only partly help with this task though, since they also have other things to do other than supervising, it is not always they have the time to read through drafts of the report or read it thoroughly. Thus it was mostly the structure of the report that was commented on rather than the content and language.

While talking about the structure of the report it is worth mentioning that there have been some minor problems with that. The main one was that the supervisor from Chalmers wanted the report to look one way while the people at Saab wanted another focus. Saab wanted to focus more on the results from the usability tests. Therefore the results from that test have been a great focus in the latter part of the report. Doing a report that way is more scientific but clashes with the focus on creating and designing an Online Help system, those report structure is a bit different with more focus on the prototype and less on the tests. But since neither side can be followed fully without important information missing the end result of the structure is compromise of the two.

As mentioned earlier some thing of this thesis were classified, this means that care had to be taken while writing the report so that none of this sensitive information was mentioned anywhere. This was not really a problem while writing the text since there was no real reason to mention whom the product was meant for nor the specs for it. The problem was the screenshots taken of the prototype, these could contain sensitive information was not immediately apparent, things like the project name that which can be used to figure out the customers country. So these pictures have to be looked through carefully by experienced people at SMW before anyone outside SMW could read the report.

14.5 The Result from the Third Usability Test

The test focused on comparing the traditional manual with the Online Help system. This may not have been the best of choices from a user-centered design perspective. Normally the usability test would have focused on finding faults with the developed product not comparing it to others, but there were two main reasons that this was not done in this case. The first was that many of the existing problems with the Online Help system was already known and those that could be fixed had already been fixed. The second reason was that SMW needed test result that they could use to show the customers to convince them that Online Help is a good compliment, and in the future a replacement, to the traditional manual. The hope was that the test would show that the Online Help was the “better” choice, and luckily it did.

During the test more effort should have been put on listening to the questions the testers asked. Too much focus was put on observing the test users action rather than listening to all they said, while taking note it was easy to miss a question or forget to write it down. Still, all in all most of the questions asked, even those that were not written down, was of general nature. But there is the possibility that an important question related to either the traditional manual or Online Help was missed and therefore could have made a difference in the result and thus could have changed the coming conclusion.

The time it took to complete the assignment differed not only between the two groups but also between the test subjects. The time deviation between the participants was probably because some people needed more time than other to get familiarized with the system, which is totally normal, the main reason these people were chosen were because of their diversities. Unfortunately this means that the results get a bit harder to interpret with such deviating numbers, fortunately the differences between the two groups clearly showed that those using the traditional manual took much longer time to

complete the assignment because it took much longer to find the right section to read, plus that the Online Help gave much clearer guidance.

There was also the odd test user that did not use the Online Help at all to solve the assignment, this means that he would probably not used the traditional manual either if he had been apart of that group. Although he did spend some time with the traditional manual when he got to try that out, but this may have been because of other issues that will be discussed shortly. This skewed the result a bit, but even if he was removed from the result the other two in the Online Help group had better completion times than the best test result in the traditional manual group.

The test users completed the most of the subtasks. Those that took longer time or had problems with certain subtasks was mostly because they did not examine the interface or read the help good enough, not because which help type they used. There were one subtask that most people failed on, it was a checkbox called “Point”, the box was checked as default but the subtask here was to understand if should be checked or not. Most did not get this even after reading the available help, giving a hint that the help (both traditional and Online) had not been elaborated enough.

Observing the way the participants used the help types assigned to them was like watching a demonstration of human nature. Men in particular had a tendency to use a trial-and-error approach to try to solve the assignment, this failed in all but one case. This seems to have something to do with the joke about that “men do not need to read manuals since they think they can figure it out anyway”. Unfortunately there is a lot of truth in this joke; men do have a tendency to not read manuals before trying out something new. This test was no different, but in the end, after failing to solve the assignment they felt obliged to start using the help in order to solve it.

Then there were the opposite; those that started with familiarize themselves with the manual or Online Help and read long sections before doing the assignment. Lastly there are those in middle that read what they are supposed to do before each subtask. The last one seems to be the one that is used most, although the test result showed that the participants were pretty much evenly divided between the three groups. This only shows that there are many different people out there, and probably far from all can be sorted into these three groups.

An interesting observation was that when the test groups got to test the other help type, i.e. the traditional manual group got to try the Online Help and vice versa, some of the participants used that help type differently than the first one. Mostly it seemed like the traditional manual was read more carefully than the Online Help, no matter which group the testers were from originally. There seems to be some rooted conception that if the information is read from the screen it is not as important as something written on paper, but this is just a theory. It could also be - at least those that started with Online Help - that because they were instructed to talk about what the differences were between the two types they felt obliged to spend extra time with the traditional manual to have something to say. This applies especially for the participant that did not use the Online Help to complete the assignment but studied the traditional manual for several minutes. So it is not certain they would have used the traditional manual that way if it had been used from the start.

There were some important differences with the two manual types; other than that one were written on paper and the other was online. The operations manual was not as thorough as the one used in the Online Help, and the part describing the creation of filters was not in the right chapter. This made it harder for those using the traditional manuals to complete the test, then again the Online Help is supposed to make it easier for the users to find the information they are looking for by combining the control and operations manuals into a single manual. There were also other parts of the interface that was not well enough explained, so both of the help types needs to be complemented and rewritten, see the next chapter for more on this.

As mentioned in the result analysis (chapter 13) one of the test users said that she preferred Online Help over the traditional manuals even though she seemed much more at ease using the traditional manuals, i.e. she studied it more carefully. This could be because she felt that she was supposed to answer Online Help and answering otherwise could be seen as retrogressive. It could also be that the question itself was formulated in a bad way, the question was “Which type of help do you generally like to use?” with the options traditional manual or Online Help. This does not ask for which of the help types she preferred in this test just which type she generally likes to use. So it is not completely out of line to say that she may have been uncomfortable using the Online Help in test, but she have no problem using Online Help for other applications.

Another thing that could have affect this result is that some of the participants knew the supervisor that assisted in the test and knew that he really likes the concept of Online Help. This could result in that they are more generous with the critic than they would otherwise be. The tester mentioned above was one of these people.

One may wonder if using numbered scales in the questionnaire is the right thing to get the test subjects thoughts about the test. It is easy to compile but it can be hard for the participants to translate their feelings to numbers. How do you really know the difference between a ‘7’ or an ‘8’? Also the ‘1’ to ‘10’ scale that was used was chosen so that there were no middle point, ‘5’ for example. This forces the person doing the questionnaire to take sides, otherwise it very easy to just choose ‘5’ which in reality really means “It is ok”. Now at least it is either a little bit good or a little bit bad. But it is not always that the person answering notices this and chooses ‘5’ anyway because he thinks that it is the middle number. This can explain that there were a lot of ‘5’ and ‘6’ answers for the group using the traditional manual.

Much of the result is a bit unclear and could perhaps have been made clearer if a larger test had been held. The main issue was the lack of participants; six participants that were divided into two groups seemed like a good idea and give a good enough result to study. But for it to be more conclusive two or three more participants per groups may have helped at lot. But since it was at the end of the vacation season at SMW it was hard to get that many people during those two days. The reason that the test was held around that time was because it was already overdue and the following week from Wednesday to Wednesday was already booked for an opposition of another group’s thesis. So in the end the test was as good as possible considering the circumstances.

14.6 Future Work

There are a lot of things left to do in the field of Online Help. As describes earlier in the report JavaHelp is far from the ideal platform of use to implement Online Help. But as also mentioned earlier, there are few others alternatives to choose from. Oracle Help can be seen as a better but more limited choice that is not as versatile as JavaHelp. So depending on what the developers want to do this can be a better choice.

Rather than depending on already existing platforms that do not fully support what the developers want to do, it may be a good idea to try to develop an Online Help system of their own. This could either be from scratch or built on an already existing system like JavaHelp. Doing this would probably take a lot of time and resources, so SMW has to decide if this is worth it. There are of course already existing solutions within Saab like the Online Help based on Firefox Portable mentioned in chapter 2.6.6 and 8.6.

The main thing that needs to be improved in JavaHelp is the window management. Pop-ups should not be limited to the size of the window that it shows in; the Pop-ups should be able to be shown outside the boundaries of their parent window. It should also be made possible for the Pop-ups to automatically change size depending on the amount of text it has to show. The current solution in JavaHelp the size has to be defined in the helpset file. This seems like a cheap solution done by Java.

There is also the issues with what currently have focus in the interface, since the users are more likely to press the F1 key after using the window they are working in for a while rather than pressing directly after opening it. Therefore it is extra important that the help content opens then the user presses the F1 key, else it will be a great source of frustration if it does not work. This can be done by either rework JavaHelp so that it recursively looks for the topic ID for the window when the F1 key is pressed, or make sure that the focus is always on the actual window and not on components.

More work needs to put down on the online manual, the topic files needs to be divided into smaller parts and the manual itself needs to be reworked to be more like an online manual and less like a traditional manual. Time should also be put into researching how RoboHelp imports FrameMaker files and see if this is a better option than importing Word documents. This was originally planned to be a part of this thesis but there were problems with downloading the latest FrameMaker version from Adobe’s server. The version currently used by SMW was too old to work with RoboHelp.

15 Conclusion

The aim of this thesis was to figure out if it was possible to implement Online Help into an already existing product in a cheap and easy way, as well as see if it was possible to create help contents with the material that already existed, i.e. not need to rewrite everything from scratch. The most suitable platform for the task, JavaHelp, was used to create several prototypes with implemented Online Help. These prototypes each went a usability test to improve the user interaction with the Online Help system. The result showed that JavaHelp while being a good choice for Online Help does not seem to be developed far enough to be used in commercial products. It is perfectly suited for presentation purposes though, to show other how Online Help could look and work. The conclusion is therefore that there is no simple way to implement a fully working Online Help in a product without having to rework and modify some parts of the help platform, in this case JavaHelp. The upside is that JavaHelp is free and does not need licensing to use.

This thesis will be used by SMW as ground for future Online Help projects, next will be a project that will test to implement Pop-ups for the use in tables and see if it can be used in a commercial product for real. The prototype itself will be used to show others how Online Help can look and work.

To use the already existing help material to create the Online Help worked well, the controls manual can easily be used to create *Pop-up* content, and by using an application like RoboHelp it is possible to quickly convert existing manuals to a working online manual. These will of course need to be modified a bit to be suitable as online manuals but most of the work has already been done and most importantly there is no need to rewrite the text to make it work. Although there were talks about combining the control and operation manuals, which of course demands that the manuals be rewritten, but that is unavoidable in that case.

The planning could have been done better, with clearer deadlines. More time should have been put on the last usability test. It was also unsuitable to do the thesis during the summer because the supervisors went on vacation in the middle of the thesis leaving me without supervising for a couple of weeks.

The term ‘Online Help’ can be misleading because it is not necessary online, but this is what this type of system is commonly referred as, so calling it something else would only be confusing.

The secrecy around the application and the intended customers complicated the supervising a bit, since the supervisor at Chalmers could not test the prototypes without visiting SMW himself. Other than that it has not been such a great issue working or writing the report, as long as nothing classified is mentioned and no compromising screenshots of the prototype is shown.

15.1 Concluding remarks on the third usability test

After taking into consideration that the questions asked during the last usability test was not of any relevant nature, and thus did not prove or disprove that Online Help was

better than the traditional manuals. This concludes that the questions did not contribute any valuable information to the result.

The result from the test showed that it took around 17 minutes per participant using the manual to complete the assignment and 8.5 for those using the Online Help, i.e. twice as fast as manual group. More participants would have been needed to make sure that this was really the case though. Still this provides a clear indication of the Online Help effectiveness compared to the traditional manuals.

The Online Help group completed 17 out of 23 subtasks and the other group 19 out of 23. The only difference is that those using the Online Help completed the subtasks faster than those that did not. This means that the use of traditional manuals or Online Help did not particularly affect how many subtasks that were completed. Based on this result it is clear that no matter how good an Online Help is it can not replace a poorly designed interface.

Each user differs from each other and therefore uses the help available differently, if at all. To create a help system that works for every one is therefore not possible, but it is possible to create a system that makes it easier for those that actually uses it.

Both of the help types need to be reworked and complemented to fully help the users understand the interface and how it works.

All the users seemed pleased with the Online Help and preferred to use it over using the old traditional manuals; at least this was what they stated afterwards.

The scores given in the questionnaire can be tied to how long it took for the participants to complete the assignment, those using Online Help had higher scores and low completion times compared to those that used the traditional manual that got low scores and had high completion time. The conclusion that can be drawn from this is that the Online Help is effective in regards to helping the user find and give the information they need to do their work.

All in all it can be concluded that more participants would have been needed in the final test to get a more conclusive result.

16 References

[Cooper 2007] Cooper, A., Reinmann, R., Cronin, D (2007). *About Face 3: The Essential of Interaction Design.* Indianapolis, Indiana: Wiley Publishing, Inc. p. 3-26, 70-71, 560-563

[Dix et al 2004] Dix, A., Finlay, J., Abowd, G. D., Beale, R (2004). *Human-Computer Interaction*. Third Edition. Madrid, Spain : Pearson Prentice Hall. p.396-414

[Ottersten et al 2008] Ottersten, I., Berndtsson, J (2008). *Användbarhet i praktiken*. China: Studendtlitteratur. p. 14-16

[ISO 13407:1999] SS-EN ISO 13407:1999 *Human centered design processes for interactive systems*. Stockholm: SIS Förlag AB.

[Eclipse 2009] Ecipse Fundation. *Eclipse Galileo Documentation.* http://help.eclipse.org/galileo/index.jsp. 2009

[ISO 9241-11] INTERNATION STANDARDS ISO 9241-11 First Edition, *Ergonomic requirements for office work with visual display terminals (VTDs). Part 11: Guidance on usability*, Ref: ISO 9241-11:1998(E), 1998-03-15. http://www.idemployee.id.tue.nl/g.w.m.rauterberg/lecturenotes/ISO9241part11.pdf

[JavaHelp 2004] Sun Microsystems, Inc. *JavaHelp 2.0 System User's Guide*. http://java.sun.com/javase/technologies/desktop/javahelp/download_binary.html. 2004-12

[Kruchten 2000] Kruchten, P. *From Waterfall to Iterative Development – A Challenging Transition for Project Managers*. The Rational Edge, 2000. http://www.ibm.com/developerworks/rational/library/content/RationalEdge/dec00/FromWaterfalltoIterativeDevelopmentDec00.pdf. 2004-12

[Mcclendon et al 1999] Mcclendon, C. M, Regot, L, Akers, G. *What is Prototyping?* http://www.umsl.edu/~sauterv/analysis/prototyping/proto.html. 1999-05-26

[Nielsen 1993] Nielsen, J. *Iterative User Interface Design*. http://www.useit.com/traditionals/iterative_design/. 1993-11

[Nielsen 2005] Nielsen, J. *How to Conduct a Heuristic Evaluation*. http://www.useit.com/traditionals/heuristic/heuristic_evaluation.html. 2005

[Oracle 2009] Oracle. *Oracle® Fusion Middleware Developer's Guide for Oracle Help 11g Release 1 (5.0).* http://download.oracle.com/docs/cd/E12839_01/doc.1111/e14149/toc.htm 2009

[Saab 2007] Saab AB. *Operations*. http://www.saabgroup.com/en/AboutSaab/Organisation/SaabMicrowaveSystems/operations.htm. 2007-06-01.

[Saab 2006] Saab AB. *Product Offerings*.
http://www.saabgroup.com/en/AboutSaab/Organisation/SaabMicrowaveSystems/product_offerings.htm. 2006-09-11

[Thomson 2009] Raynald Thomson Ltd. *What Is Software Prototyping?*
http://www.reynardthomson.com/what-is-prototyping.html. 2009

[UPA 2005] Usability Professionals' Association (UPA). *Heuristic Evaluation*.
http://www.usabilitybok.org/methods/p275. 2005

[UsabilityNet 2006] Usability Net. *Design Guidelines*.
http://www.usabilitynet.org/tools/designguidelines.htm. 2006

Appendix A – Evaluation of Usability Test 1

Positiva noteringar	**Klassificering**	**# pers**
Generellt mycket hög acceptans på det totala konceptet	-	3
Att få upp Popup fönster med ett musklick.	Interaktion	3
Att få kort snabb hjälp med vissa komponenter.	Effektivitet	3
Att kunna få hjälp via klick på label och inte bara på aktuell komponent	Effektivitet / tillfredställelse	3
Att nå popup med mittenknappen	Interaktion	1
Bra att fönsterplacering sker rakt under markören	Fönsterplacering	1
Negativa noteringar		
Inte går att förstora/minska Popup.	Fönsterhantering	2
Inte går att flytta Popup (kan ibland dölja GUI).	Fönsterhantering	1
Att inte kunna använda högermusknapp för att öppna Popup	Interaktion	1
Scrollbar har lägre acceptans än större storlek på popup	Fönsterhantering	1
Fönsterplaceringen gjorde ibland att labels skymdes	Fönsterplacering	1
Att inte markerat objekt highlightas i F1-dialog när den öppnas.	Effektivitet / Tillfredställelse	1
Övriga kommentarer		
Om popuperna kan flyttas inom en och samma dialog, så skall den minnas positionen. Vid öppnande av en ny skall den mao öppnas på samma plats som föregående.	Fönsterplacering	1
Att som val kunna lägga en separat hjälpdialog i sidopanelen som hela tiden visar hjälptext för den komponent man för tillfället arbetar med	Ändamålsenlighet	2
Från Popup fönstret kunna länka sig vidare till t.ex. ordlista (för komplicerade begrepp eller förkortningar) eller annan lämplig text i manualen. Denna skulle i så fall länkas till akutellt kapitel i F1-dialogen.	Ändamålsenlighet	1
Dynamisk text som förklarar de nuvarande inställningarna.	Ändamålsenlighet	1

Klassificering (endast positiva) inkl # personer	**Antal**	**Kommentar**
Interaktion	4	Enkel interaktion med mycket hög acceptans
Ändamålsenlighet	0	-
Effektivitet	6	Snabb åtkomst och bidrog till hög effektivitet och tillfredställelse.
Tillfredställelse	3	Se ovan
Fönsterplacering & Fönsterhantering	1	-

Klassificering (ej positiva) inkl # personer	Antal	Kommentar	Tidsåtgång
Interaktion	1	-	
Ändamålsenlighet	4	Funktionsaddering är av lägre prioritet i detta skede av utvecklingen. Noteringar tas med vid nästa fas.	3
Effektivitet	1	-	
Tillfredställelse	1	-	
Fönsterplacering	6	Flera av kommentarerna hanterar detta ämne och bör därmed prioriteras i fortsatta frågeställningar.	1
Fönsterhantering	6	Se ovan	4

Appendix B – Evaluation of Usability Test 2

Positiva noteringar	**Klassificering**	**# pers**
Generellt mycket hög acceptans på det totala konceptet	-	3
Att få upp Popup fönster med ett musklick.	Interaktion	3
Att få kort snabb hjälp med vissa komponenter.	Effektivitet	3
Att nå popup med mittenknappen	Interaktion	2
Att nå popup med högerknappen	Interaktion	1
Lägger inte märke till de olika storlekarna	Fönsterhantering	3
Bra att fönsterplacering sker under komponent	Fönsterplacering	1
Negativa noteringar		
Inte går att flytta Popup	Fönsterhantering	1
Texten stämmer inte/Rubriker behövs ändras	Manual	2
Manual inte tillräckligt utförlig, behöver mer hjälp än vad vissa knappar mm betyder	Manual	1
Ser inte poängen med ”What is this?” menyn/ Känns lite konstigt med bara ett menyalternativ	Interaktion	2
Vissa komponenter ska inte behöva Popup hjälp, ska vara självförklarande	Ändamålsenlighet	1
Ingen ?-knapp (Windows standard)	Interaktion/ Tillfredställelse	2
Fokus på F1 fungerar inte alltid	Interaktion/ Effektivitet	3
Ingen hjälp för mål på kartan	Interaktion/ Tillfredställelse	1
Ingen hjälp i menyer	Interaktion	1
Övriga kommentarer		
Behövs felsökningsschema, varför fungerar det inte som det ska etc Wizards kanske är en lösning	Ändamålsenlighet	1
Tryck på F2 för att få fokus på Popup (som i Eclipse), sedan kunna flytta och ändra storlek mm	Ändamålsenlighet	1
Länkar i manualen för att knyta ihop olika funktioner mm	Manual/ Ändamålsenlighet	1

Klassificering (endast positiva) inkl # personer	Antal	Kommentar
Interaktion	6	Enkel interaktion med mycket hög acceptans
Ändamålsenlighet	0	-
Effektivitet	3	Snabb åtkomst
Tillfredställelse	0	-
Fönsterplacering & Fönsterhantering	4	Bra respons på placering och storlek på Popups

Klassificering (ej positiva) inkl # personer	Antal	Kommentar	Tidsåtgång 1=lätt fixat 5=dream on
Interaktion	9	Små grejer, inte så viktiga	3
Ändamålsenlighet	1	Bättre interface, mindre Popups	5
Effektivitet	3	Fixa fokusproblem vid F1	3
Tillfredställelse	3	-	-
Manual	3 - 4	Mesta fixas med en mer genomarbetat manual	-
Fönsterplacering	0	-	-
Fönsterhantering	1	-	4

Appendix C – Usability Test Assignment

Introduktion
Testet görs för att jämföra användningen av en onlinehjälp mot en pappersvariant av användarmanualen. Du har blivit ombedd att göra testet med hjälp av:

___ pappersmanual ___ onlinemanual

Om funktionen TDN
Testet behandlar funktionen TDN-filter (TDN = Tactical Data Net - nätverket för utbyte av data mellan olika C2-enheter). Ett TDN-filters uppgift är att *reducera utgående data* och detta görs i regel pga att utgående information till en annan C2-enhet är överlastat. Mha filtret kan man därmed få ner den utgående informationen.

Scenario
Du har upptäckt att en av era flygplatser genererar en stor mängd luftmål, vars information dina andra C2-enheter inte är intresserad av och som dessutom är på väg att överlasta nätet. Du vill därmed upprätta ett filter kring denna flygplats.

Uppgift
Din uppgift är att upprätta ett TDN-filter som
- Är aktivt runt flygplatsen
- Får bort information om dina egna flygenheter då de startar och landar.
- Tillser att du fortfarande kan se mål som flyger på hög höjd ovanför flygplatsen

Du når TDN-funktionen från huvudmenyvalet **Communication.**

Hjälp
I första hand skall du använda dig av den/de manual/-er som finns tillgänliga. Som sista utväg får du be observatören om hjälp. Tänk gärna högt.

Totalupplevelse
När du genomfört uppgiften kommer du att få prova att göra samma sak med den andra manualtypen. Även om du nu vet hur man gör så får du försöka att se det ur perspektivet att du inte vet. Det går bra att ”klämma och känna” i högre utsträckning, för att sedan ge ett omdöme om ditt totala intryck.

Appendix D – Observation protocol

___ Pappersmanual ___Onlinemanual

Händelse	**Antal**	**Kommentar**
Antal tillfrågningar till observatör		
Följande aspekter är uppfyllda i genomförandet av uppgiften	-	1. ___ Aktivera filtret 2. ___ Förstå hur Point fungerar 3.___ Välja rätt Plattform 4. ___ Välja rätt identiteter 5. ___ Införa geografisk begränsning 6___ Sätta höjdbegränsning 7. ___ Filtrera rätt sida av begränsningen
Intressanta kommentarer/reflektioner från användaren under testets gång	-	
Observatörens spontana känsla av skillnad vid användning av andra hjälptypen	-	
Hur löses uppgiften, sett ur perspektivet på vilket sätt användaren nyttjar tillgänglig hjälp	-	___ Sekventiellt med infohämtning allt eftersom – trial and error ___ Läser först igenom allt, får förståelse och agerar därefter ___ Hybrid, läser delar, får förståelse, agerar och gör samma sak om igen

Du har fått i uppgift att använda:
__ pappersmanual
__ onlinemanual

Appendix E – Questionnaire

1. Hur upplevde du tidsåtgången för att hitta *rätt information*?

Mkt tidskrävande Mkt tidseffektivt

|_____|

1 2 3 4 5 6 7 8 9 10

2. Hur upplevde du kvaliteten på informationen som sådan?

Intetsägande Mkt givande

|_____|

1 2 3 4 5 6 7 8 9 10

3. Till vilken grad upplevde du att du fick hjälp att nå ditt mål?

Mkt liten grad Mkt hög grad

|_____|

1 2 3 4 5 6 7 8 9 10

4. Hur lätt är hjälpen att lära sig att använda (på ett effektivt sätt)?

Mkt svårt Mkt lätt

|_____|

1 2 3 4 5 6 7 8 9 10

5. I vilken grad upplever du att hjälpen ger information om vilka steg som är nödvändiga att genomföra och varför de är viktiga att genomföra för att nå ett visst mål?

Mkt liten grad Mkt hög grad

|_____|

1 2 3 4 5 6 7 8 9 10

6. I vilken grad upplever du att hjälpen är navigerbar (”var är jag?”)?

Mkt liten grad Mkt hög grad

|_____|

1 2 3 4 5 6 7 8 9 10

Totalupplevelse

Efter att du genomförde testet fick du prova samma förfarande med den andra hjälpvarianten. Efter att ha fått kännedom om de båda så får du svara på följande tre frågor:

1. Över lag – vilken typ av hjälp föredrar du?

___ Användarmanual / operationskapitel ___ Onlinehjälp

2. Vad upplever du är den största fördelen (/-arna) av att använda den typ av hjälp du föredrar?

_____

_____

_____

3. Övriga tillägg?

_____

_____

_____

_____

_____

_____

Appendix F – Results from Usability Test 3

Questionnaire Answers

Traditional manuals

Question 1	1	2	3	4	5	6	7	8	9	10	Numerical
#1								X			8
#2			X								3
#3				X							4
mean val.					**X**						**5**
stand. dev.											2,6457513
Question 2	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
#1				X							4
#2						X					6
#3					X						5
mean val.					**X**						**5**
stand. dev.											1
Question 3	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
#1				X							4
#2					X						5
#3						X					6
mean val.					**X**						**5**
stand. dev.											1
Question 4	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
#1				X							4
#2				X							4
#3				X							4
mean val.				**X**							**4**
stand. dev.											0
Question 5	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
#1				X							4
#2		X									2
#3					X						5
mean val.			**X**	**X**							**3,6666667**
stand. dev.											1,5275252
Question 6	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
#1				X							4
#2						X					6
#3					X						5
mean val.					**X**						**5**
stand. dev.											1

Online Help

Question 1	1	2	3	4	5	6	7	8	9	10	Numerical
#1									X		9
#2							X				7
#3								X			8
mean val.								**X**			**8**
stand. dev.											1
Question 2	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
#1								X			8

#2								X			8
#3						X					6
mean val.							**X**				**7,3333333**
stand. dev.											1,1547005
Question 3	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
#1									X		9
#2								X			8
#3							X				7
mean val.								**X**			**8**
stand. dev.											1
Question 4	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
#1									X		9
#2									X		9
#3									X		9
mean val.									**X**		**9**
stand. dev.											0
Question 5	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
#1									X		9
#2							X				7
#3						X					6
mean val.							**X**	**X**			**7,3333333**
stand. dev.											1,5275252
Question 6	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
#1									X		9
#2								X			8
#3							X				7
mean val.								**X**			**8**
stand. dev.											1

Participant thoughts

What kind of help did you prefer?

All six participants preferred the Online Help over the traditional manuals.

What do you think are the biggest advantages with the kind of help you prefer?

(Since all user answered Online Help on the previous question that will be the preferred help type)

It's more detailed.
It is easier to see in which order you have to do stuff.
There is a search function available.
Quicker explanation of buttons and fields.
Easier to find what you look for.
Direct access to information about the component/function that you are currently using.

Other thought

People should be educated in the use of manual and then practise with the Online Help in order to get a feeling of what kind of functions that exist and then quickly be able get hold of details through the Online Help.

It is a bit annoying that the Online Help "disappear" every time you click on something in the interface. You want to be able to have the Online Help open beside the interface

while working. Instead there is a lot of clicking every time you want to open the Online Help again.

The Online Help works very well when needing information about a certain field, i.e. short brief information. But it the user manual feels like a better choice if you want to read longer sections to understand how a function works.

No matter how good the Online Help is it is still important that the interface is as intuitive as possible so that you won't need it.

Results from the observation protocols

Traditional manuals

Tester	Time (min)	# Questions	1	2	3	4	5	6	7	Type 1	Type 2	Type 3
#1	22	3	X	X	X		X	X*		X		
#2	16	2	X		X*[2]	X	X	X	X	X*[3]		X*[4]
#3	13	1	X		X	X	X	X	X		X	
∑		6							17			

* Wrong values *[2] After hint *[3] 1st half *[4] 2nd half

Mean value time: 17 min
Standard deviation: 4.58257569

Observations:

#1 Took a while to find 'New TDN Filter', does not read to get help
Want to have drag function in Sit-display.
Trial-and-error all the way. Does not even read the labels.

#2 Have problems to create geographic boundaries because of EW-strobes. Want to have warning text for this.
Thought you needed to create the geographic boundaries in a separate window other than TDN filter. Goes into 'Shapes, symbols and areas' lo look for it.
Should not have EW-strobes because you cannot make filter for them.

#3 Looks for filter in the chapter for the L11B window in the operation manual.
Have problems understanding why you cannot make geographic boundaries for EW-strobes but manage to figure it out.
Uses drag and drop in Sit-display
Unsure about 'Point'.

Differences between Online Help/traditional manuals

#1 Likes the search function. Much easier to use, likes Pop-ups.
Thinks that the font in the Online Help should be the same as the font in the rest of the application.

#2 Uses the search to great extension.
Does not use Pop-ups unless told to do so.
Have the same problem understanding EW strobes.

#3 Quicker to get where you want. Better descriptions/clear procedures. Likes Pop-ups.
Too much "programmer language". Hard to get an answer to why you have to do a particular task to get something to work.

Previous experiences/background

#1 No earlier experiences of similar systems.

#2 Not much, do not know what TDN & TDUN is.

#3 Verified C2 system Unde23. There are no filters in this system.

Online Help

Tester	Time (min)	# Questions	1	2	3	4	5	6	7	Type 1	Type 2	Type 3
#4	9.75	0	X	X*	X	X	X	X	X		X*[2]	
#5	13	2	X		X*[3]		X	X	X			X
#6	4	0	X	X*[4]	X	X	X*[5]	X	X	X		
∑		2							19			

* Made a decision, but have not really understood *[2] Pop-up first then manual. Read first act later. Seeking understanding *[3] Needed hint *[4] Not until breifing *[5] Very narrow

Mean value time: 18.5 min
Standard deviation: 6.36396103

Observations:

#4 Want to drag and drop directly into the Sit-display.
Found L11B's chapter about filters, uses it instead.
Annoying that the Online manual 'disappears' every time you click on something in the user interface.
Tries to open Pop-up in coord. field, does not try to do the same thing on the label.
Right reasoning with 'Point', still unsure what really means.

#5 Filter inside/outside hard to understand.
Point feels superior to platform/ID in GUI. Did not use the Online Help to understand it.
Thinks that it should say what is needed for the options in the group box to work.
It is not apparent why you can't set geographical boundaries. Not clearly stated in the manual.
Hardly uses Pop-ups.
Does not think that 'New…' is associated with filter, should say 'New Filter…' instead.

#6 Hardly used the Online Help.
100% reasoning from reading labels and trial-and-error.
Started reading the online manual when he noticed 'Point', but did not understand if it should be checked or not.
Thinks that Pop-ups seems to be a good idea if you are uncertain of what a component means. Although there are not Pop-ups for all components nor do they always help.
An interface should be intuitive enough to not need help.

Differences between Online Help/traditional manuals

#4 Hard to see the relations between the operations and controls manuals (this can be because they are written by two different people).
Takes longer to look something up in the table of contents. Generally takes a longer time.

#5 Easier to see that 'New…' is associated with filter.
Thinks that some parts are explained more clearly, are not sure if the same was explained in the online manual or not though.
Feels more at ease with the traditional manuals and read it more thoroughly than the online manual.

#6 Does not find filter easily in the Op. manual. Because it is placed under L11B you do not connect that it is the same thing.
Control manual gives enough help if you are uncertain.
Checks the papaer manual more thoroughly, probably becasue he was told to talk about the differences between the two manual types.

Previous experiences/background

#4 No experience with HMI about TDN. Works with 'stuff' at a deeper level.
#5 Worked with UndeE-HMI. No TDN filter.
#6 System tester. Nothing about TDN.