

MSc. Thesis in Interaction Design

# SpaceLab Usability Evaluation - Recommended Design Improvements

Johan Ågren

Göteborg, Sweden 2006



IT University  
of Göteborg

CHALMERS | GÖTEBORGS UNIVERSITET

Department of Computer Science





REPORT NO. 2006:18

# SpaceLab Usability Evaluation - Recommended Design Improvements

JOHAN ÅGREN



Department of Computer Science  
IT UNIVERSITY OF GÖTEBORG  
GÖTEBORG UNIVERSITY AND CHALMERS UNIVERSITY OF  
TECHNOLOGY  
Göteborg, Sweden 2006

SpaceLab Usability Evaluation -  
Recommended Design Improvements  
Johan Ågren

© JOHAN ÅGREN, 2006

Report no 2006:18  
ISSN: 1651-4769  
Department of Computer Science  
IT University of Göteborg  
Göteborg University and Chalmers University of Technology  
P O Box 8718  
SE – 402 75 Göteborg  
Sweden  
Telephone + 46 (0)31-772 4895

[Matematiskt Centrum]  
Göteborg, Sweden 2006

# **SpaceLab Usability Evaluation - Recommended Design Improvements**

JOHAN ÅGREN

Department of Computer Science

IT University of Göteborg

Göteborg University and Chalmers University of Technology

## **Abstract**

This thesis is written with the purpose to evaluate the software SpaceLab, developed by Carmenta AB, from a usability point of view. The main goal was to find and document usability problems and based on these findings; recommend a number of design improvements to enhance the usability in the application. By studying relevant literature related knowledge was acquired. Background information to the system was provided in an interview held with a support worker. Heuristical evaluation was conducted with individual interaction designers as experts followed by usability tests with potential system users as test subjects. The findings from these different parts were then analysed and resulted in a number of recommendations for SpaceLab design alterations. The recommendations concerned for instance the way feedback is presented, the lack of a tutorial introduction and the lack of consistency within the software.

**Keywords:** SpaceLab, cognitive psychology, heuristical evaluation, heuristics, usability, human computer interaction, usability test.

## Acknowledgements

The process of writing this master thesis has been both challenging and developing, and I can honestly say that it has been a fun journey. But I could not have completed it on my own and therefore there are some persons who really deserve my gratitude. Thank you Tobias Moberg, my supervisor at Carmenta, for support and good advice. Also my tutor at Chalmers, Morten Fjeld, receives my gratitude. And to all persons involved in any part of the study, this thesis would not exist without your contribution – thank you. I would also like to say special thanks to my sister, Lotta Lindberg who has given me both ideas of how to conduct the study and moral support when I needed it.

Johan Ågren  
Gothenburg, January 2006

# Table of Contents

<b>CHAPTER 1 - INTRODUCTION</b> .....	1
1.1 Background .....	1
1.2 Carmenta – organizational overview .....	1
1.3 Problem area and delimitations of the study .....	2
1.4 Scope of the thesis .....	2
1.5 Purpose of the thesis – research questions .....	3
1.6 Thesis overview – chapter introduction .....	4
1.7 Software introduction .....	4
1.7.1 <i>Spatial Ace</i> .....	5
1.7.2 <i>Relation between SpatialAce and SpaceLab</i> .....	5
1.7.3 <i>SpaceLab</i> .....	6
<b>CHAPTER 2 – THEORETICAL RESEARCH FRAMEWORK</b> .....	9
2.1 Cognitive psychology .....	9
2.1.1 <i>Information processing</i> .....	10
2.1.2 <i>Perception</i> .....	10
2.1.3 <i>Perceptual organisation</i> .....	11
2.1.4 <i>Attention</i> .....	11
2.1.5 <i>Pattern recognition</i> .....	12
2.1.6 <i>Mental models</i> .....	13
2.1.7 <i>Feedback</i> .....	14
2.1.8 <i>Colour</i> .....	14
2.1.9 <i>Coding</i> .....	15
2.2 HCI – Human Computer Interaction .....	16
2.3 Interaction Design .....	17
2.4 Usability .....	17
2.6 Usability testing - definition .....	19
2.7 Heuristic evaluation .....	19
2.8 Usability test .....	20
2.8.1 <i>Test plan</i> .....	20
2.8.2 <i>Test participants</i> .....	21
2.8.3 <i>Constructing test</i> .....	22
2.8.4 <i>Walkthrough and Pilot test</i> .....	22
2.8.5 <i>How to conduct the test</i> .....	23
2.8.6 <i>Think-aloud protocols</i> .....	24
2.8.7 <i>Structuring the results</i> .....	24
2.9 Methodology Theory .....	25
2.9.1 <i>Literature studies</i> .....	25
2.9.2 <i>Approaches of research</i> .....	25
2.9.3 <i>Interviews</i> .....	25
2.9.4 <i>Questionnaire</i> .....	26
2.9.5 <i>Analysis</i> .....	27
2.9.6 <i>Validity and Reliability</i> .....	27

**CHAPTER 3 – THE EMPIRICAL STUDY.....29**

3.1 Literature study.....29

3.2 Feasibility study.....30

    3.2.1 *Interview with SpatialAce support employee* .....30

    3.2.2 *Heuristic Evaluation* .....30

3.3 Usability Test .....32

    3.3.1 *Test plan* .....32

    3.3.2 *Choosing test participants*.....33

    3.3.3 *Constructing the test* .....33

    3.3.4 *Conducting the usability test* .....34

    3.3.5 *Structuring and analysing the results* .....35

3.4 Results of the empirical study .....37

    3.4.1 *Interview*.....37

    3.4.2 *Heuristic evaluation* .....38

    3.4.3 *Usability Test*.....42

    3.4.4 *Post test questionnaire* .....51

    3.4.5 *Result summary*.....53

**CHAPTER 4 – DISCUSSION, RECOMMENDATIONS AND CONCLUSION 54**

4.1 Discussion .....54

    4.1.1 *Research process* .....55

    4.1.2 *Empirical study*.....58

    4.1.3 *Further remarks*.....63

4.2 Recommendations .....63

    4.2.1 *Design recommendations* .....63

    4.2.2. *Overall recommendations* .....65

4.3 Conclusion.....65

4.4 Future research .....66

**REFERENCES.....67**

- Appendix 1 - Usability Test Planning**
- Appendix 2 - Heuristic Evaluation**
- Appendix 3 - Testmanus**
- Appendix 4 - Medgivande till videoupptagning**
- Appendix 5 - Avslutningsenkät**
- Appendix 6 - Kick Ass Curve**

## Table of figures

Figure 1. Relation between SpatialAce and SpaceLab.....	6
Figure 2. SpaceLab visual overview .....	7
Figure 3. Changing colour attribute in the object editor .....	7
Figure 4. Index pane - Classes tab active .....	8
Figure 5. Heuristics for expert evaluations.....	32
Figure 6. A description of the analysis process .....	36
Figure 7. The Yardstick™ heuristics.....	36
Figure 8. Example of complicated operator names.....	39
Figure 9. The object shelf .....	41
Figure 10. Icons for node size changing.....	41
Figure 11. Index tab in help section .....	44
Figure 12. List of unsorted reference systems .....	45
Figure 13. Incomplete node coded in green.....	46
Figure 14. Undefined dataset .....	46
Figure 15. Editor for attribute depended visualisation.....	47
Figure 16. Buttons to open editors .....	48
Figure 17. RenderOp with two colour objects.....	49
Figure 18. Key table code.....	49
Figure 19. Clarification of the location of an error .....	50
Figure 20. A node with a comment .....	53
Figure 21. Level of agreement: SpaceLab is easy to use .....	61
Figure 22. Level of agreement: It is easy to get lost in SpaceLab .....	61
Figure 23. Level of agreement: Feedback is sufficient in SpaceLab .....	62

## Table of tables

Table 1. Summary of users' impression of SpaceLab.....	51
Table 2. Summary of programmers' impression of SpaceLab .....	52
Table 3. Summary of non-programmers' impression of SpaceLab .....	52

# CHAPTER 1

## INTRODUCTION

---

*This chapter includes background information about the thesis and the company Carmenta. It also describes the problem area and delimitations of the study along with its purpose and the parts that are included in the report. It ends by introducing the software SpatialAce and its visual tool SpaceLab and the way they are connected.*

---

### **1.1 Background**

This master thesis is carried out at the Master program in Human Computer Interaction – Interaction Design at the IT University in Gothenburg, Sweden. During the master program different approaches to interaction design and usability have been presented, among these were usability tests and evaluations. These were also the two areas found most interesting by the author to this thesis. To be able to practice these interests a contact with Carmenta AB (later known only as Carmenta) was made. During a few meetings the subject was discussed and it turned out that Carmenta needed a usability evaluation of their product SpatialAce.

### **1.2 Carmenta – organizational overview**

The company Carmenta was founded in Gothenburg 1985, basing their business on consultancy with a primary focus on software development tools. Five years later the primary focus changed to involve the defence and aviation industry. In this area Carmenta have been involved in the development of a number of advanced systems including C3I (Command, Control, Communication and Intelligence) applications and simulators.

In 1991 the company started its research on GIS (Geographical Information Systems) technology, and this research led to the development of the SpatialAce mapping toolkit, of which a part will be the main focus of this thesis.

Carmenta today focus on products and services that can be used to build mission critical systems based on command and control as well as GIS technology. These products are mostly used by system integrators and software/system developers within defence, aviation and public safety sectors. Carmenta currently employs around 50 people located in three different offices, Gothenburg, Stockholm and Borås.

### ***1.3 Problem area and delimitations of the study***

SpatialAce was originally developed to serve as a development tool inside the Carmenta organisation, to help them create applications that visualize geographical data. It has been developed from an application based totally on text input to a graphical user interface toolkit used when working with GIS. During the development, SpatialAce has also changed from only being a developer's tool at Carmenta to also serving as a product to external customers. That transformation into a product on a customer market brought a number of new demands to SpatialAce, which have been fulfilled in different new releases of the software. Despite these updates in the new releases, no usability study has ever been done regarding how potential users of SpatialAce perceive the product. Carmenta thought that such a study could benefit the continuous development of the application and agreed with the author to perform such a study.

In order to keep the thesis within reasonable width some delimitations of the research area has been made. The thesis only discusses the part of SpatialAce called SpaceLab where the configuration files are created and configured. As Carmenta wanted to explore how new users perceive and get started with the product; that too came to work as a boundary to the research area. These new users should partly have programming experience and partly not, in order to see if there was any noticeable difference in how those groups perceived the interface. Apart from the usability test including new users, an expert evaluation of the interface was conducted to broaden the ground for the usability change recommendations. These limits conclude that the thesis handles how new users perceive and get started with SpaceLab along with an expert usability evaluation of the interface. Another important limitation is the fact that the recommendations will only include information about where problems can be found, and not real visual design suggestions since that would undertake too much time to complete.

### ***1.4 Scope of the thesis***

The thesis targets Carmenta as a corporation but might also be found useful to others with an interest in interaction design in general and application usability evaluations in specific. No explicit knowledge of specific terminology is required but a certain basic knowledge of human computer interaction and some recognition of programming related terminology will ease the process of fully understanding the aspect of the thesis.

The study was carried out in a few different phases; during the first phase a theoretical study was conducted to gain a deepened understanding of what constitutes usability, more knowledge of usability testing and an understanding of SpaceLab as a tool.

Secondly, a feasibility study was conducted. This study included a short unstructured interview with a SpatialAce support worker. The aim of that interview was to get a first impression of what usability flaws in SpaceLab that already might be known of and see how they were handled. The feasibility study also included a heuristic evaluation of SpaceLab carried out by experts in the interaction design area. This evaluation was conducted to find usability flaws that might not be detected in later phases, but also to serve as background material when constructing the scenarios for the next phase.

The third phase included a scenario-based usability test of SpaceLab. The purpose of the test was to collect data in order to obtain results that would serve as a basis for the recommendations for a SpaceLab usability adjustment.

The last phase consisted of analysis of test results and a discussion about the results in relation to relevant theoretical research combined with outlining recommendations for a SpaceLab usability modification.

### ***1.5 Purpose of the thesis – research questions***

The thesis is based on an assignment provided by Carmenta and aims at detecting and documenting usability problems within the product SpatialAce and in particular its visual tool SpaceLab. It is believed that a number of usability flaws will be found in the software by using different methods such as literature studies, interviews, heuristical evaluations and usability tests. The found usability problems will be viewed and analyzed in order to create suggestions of improvements. Therefore, the research questions of this thesis can be formulated:

- What problems do new users to SpaceLab experience and how can those problems be reduced?
- What are the overall usability problems within SpaceLab and how can they be reduced?

Of lesser importance, but something that would be good to find an answer to is:

- Is there a difference in how new users with programming experience understand SpaceLab compared to new users without programming experience? What is that difference?

The result of the study should consist of a number of recommendations for a design alteration of SpaceLab in order to make the product more usable. These design suggestions should agree with human cognitive characteristics as well as relevant design guidelines.

### **1.6 Thesis overview – chapter introduction**

#### **Chapter 1**

*Includes background information about the thesis and the company Carmenta. It also describes the problem area and delimitations of the study along with its purpose and the parts that are included in the report. It ends by introducing the software SpatialAce and its visual tool SpaceLab and the way they are connected.*

#### **Chapter 2**

*Deals with the theoretical research framework for the thesis by presenting concepts relevant for the study. It starts with a review of human cognitive limitations together with information about coding, colour and feedback. Thereafter comes an introduction to basic HCI, interaction design and usability concepts and the chapter continues with a thorough introduction to usability testing. The last part concerns methodology theory.*

#### **Chapter 3**

*Describes how the empirical study was conducted and which results that was obtained. It explains how the three different phases of the study was performed, the literature study, the feasibility study and the usability test. The first part of the chapter describes how each phase was designed and the second part of the chapter describes the overall findings of each phase. These findings will be discussed in chapter four in relation to relevant theory acquired in chapter two.*

#### **Chapter 4**

*Connects the different parts of the study and gives possible answers to the research questions found in the first chapter. The results acquired in the third chapter is interpreted and related to the theoretical research framework. Also included is a discussion about the research process with notions on what have been successful and what could have been done differently. Following the discussions are the recommendations for design alterations. The chapter ends with conclusions and recommendations for future research.*

### **1.7 Software introduction**

To give the reader an idea of what software is being evaluated the coming sections will introduce SpatialAce in general and SpaceLab in particular. Information and pictures comes from the help section within each program.

### 1.7.1 Spatial Ace

SpatialAce is a general configurable tool for presentation of maps, working with geographical information and interaction with geographical objects in two as well as in three dimensions. The system is developed to be built into other applications, making geographical components a fully integrated part of the application.

SpatialAce consists of several different parts:

- The base of SpatialAce components can handle reading, presentation and manipulation of geographical data, and among these parts you will find: A set of components whose responsibility is to handle reading and/or writing a large number of geographical data formats, flexible attribute-controlled visualization of the geographical data sets and a plug-in which facilitates visualization of geographical data in three dimensions.
- Different programming interfaces (APIs) that makes it easy to use SpatialAce from a number of programming languages. These include: a complete COM-interface, .NET Primary Intertop Assemblies, a Java interface and the script language Pilsner that are mostly used in the map configuration files.
- SpaceLab – a visual tool for creating and editing the configuration files in SpatialAce. These files control reading and visualization of the geographical data.
- SpaceExplorer – a stand-alone viewer for the configuration files.

### 1.7.2 Relation between SpatialAce and SpaceLab

The core purpose of SpatialAce is to create maps and geographical components that can be integrated in other applications. Figure 1 tries to explain the relations between the different parts of the SpatialAce tool. In centre of all relations is the .p file, which is the configuration file. This file is created in SpaceLab and can always be opened there to be edited, as it can be opened in SpaceExplorer to present a preview of the map created. It is also possible to initiate this command from SpaceLab so the users can preview their creations when altering the object attributes. The map created can then be integrated into an extern application by including a SpatialAce object within the software, with a reference to the configuration file.

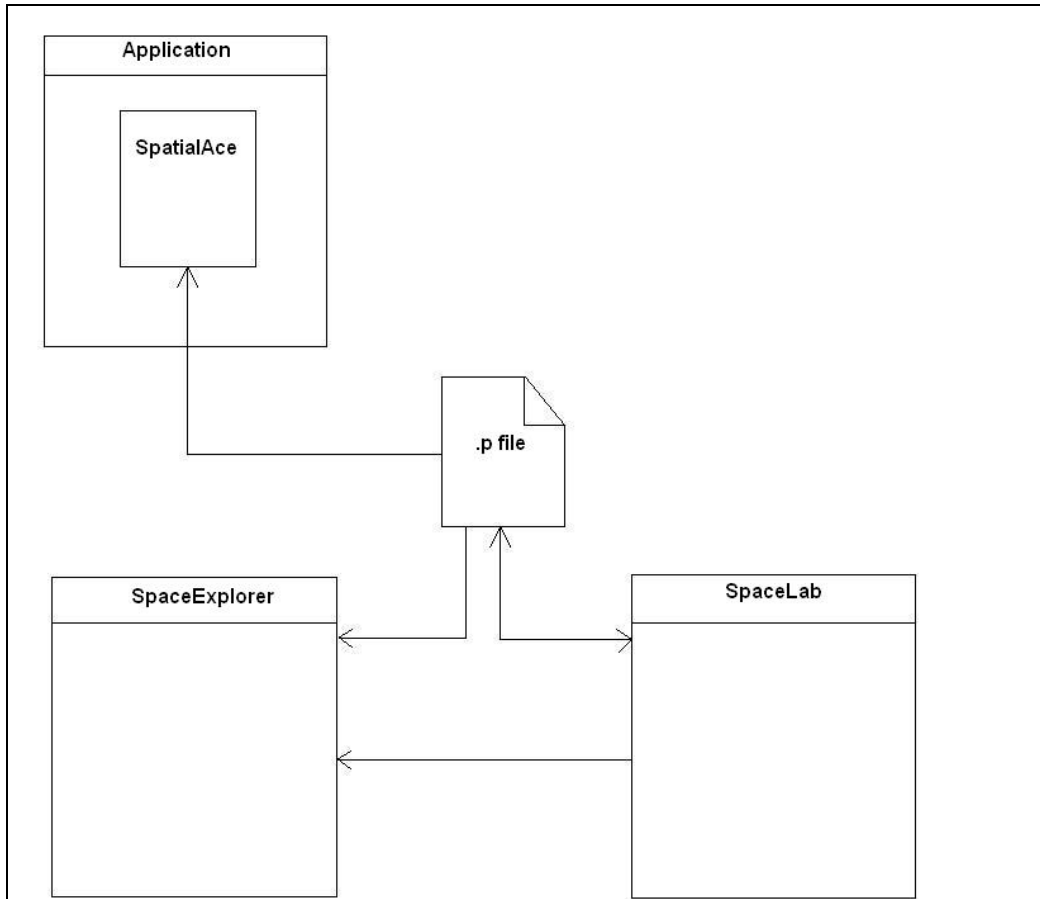


Figure 1. Relation between SpatialAce and SpaceLab

### 1.7.3 SpaceLab

SpaceLab is, as stated above, where the configuration files for SpatialAce are created. It describes what data that should be read, how it should be processed and in what way the data should be visualised. The appearance of SpaceLab is divided into three areas, apart from the menu bar and tool panel; the index pane to the left, the object editor area below the index pane, and the dataflow window to the right.

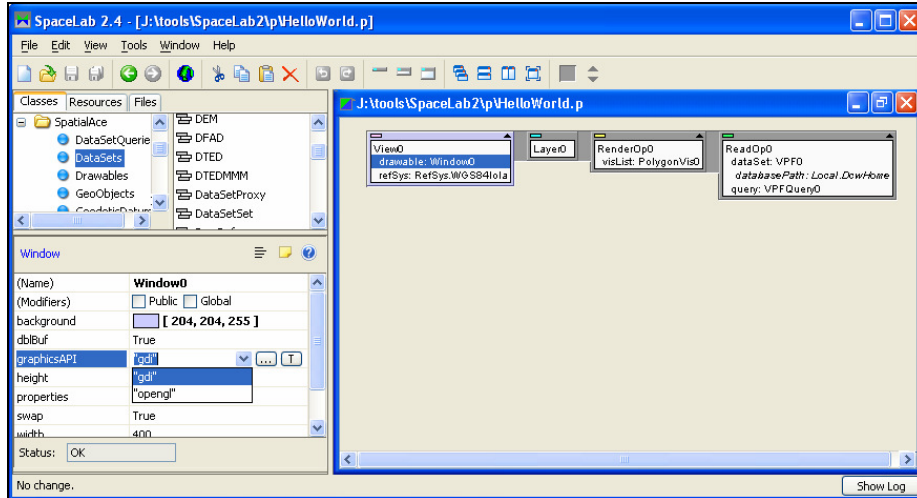


Figure 2. SpaceLab visual overview

The configuration files in the right hand window are structured and displayed as a dataflow diagram, to make it easier to grasp. To preview the configuration files in SpaceExplorer the user has to press the button marked with a globe in the toolbar, or choosing the SpaceExplorer option from the tool menu.

To add or move objects to the configuration file, SpaceLab relies on drag and drop handling. The user selects the wanted object in the class library and simply drags it to the appropriate place and drops it there. When a user wants to change an attribute in the object editor he selects the appropriate row and can thereafter choose from a few different options. Either he can change the attribute by typing the wanted value on the row, or if applicable he can choose the wanted value from a drop down list, or he can change the value from one of the two possible editors – custom or text. In figure 2 the colour row is selected, presenting the user with the mentioned options. In this case all alternatives are possible, the drop down menu represented by an arrow pointing down, custom editor symbolized by three dots and the text editor shown as a button labelled T.

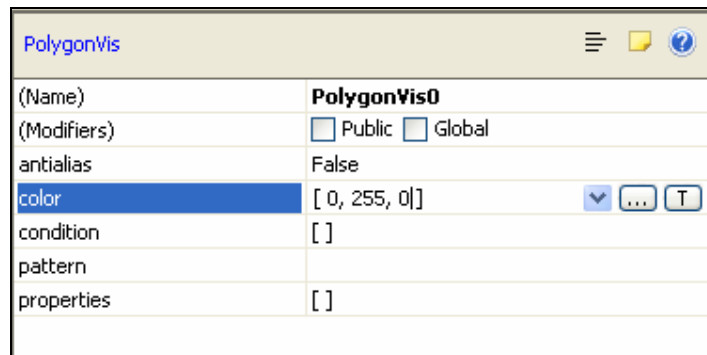


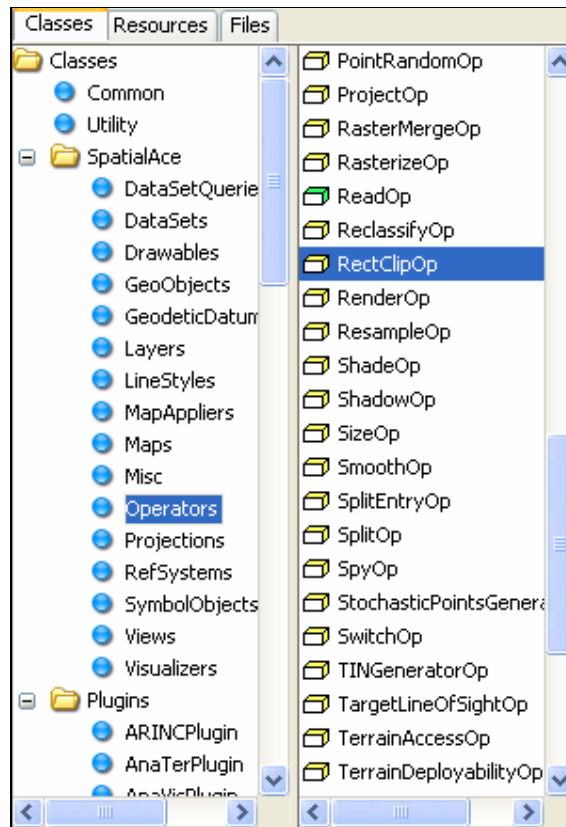
Figure 3. Changing colour attribute in the object editor

The index pane is divided into three tabs: Classes, Resources and Files. In the class tab all classes and objects needed to create new objects instances. It is constituted by a

## INTRODUCTION

---

tree and a list, the list showing all objects in the selected folder in alphabetical order. The tab Resources provides access to all known system resources, and Files shows a list of objects used in the currently active configuration file.



**Figure 4.** Index pane - Classes tab active

## CHAPTER 2

### THEORETICAL RESEARCH FRAMEWORK

---

*Chapter 2 deals with the theoretical research framework for the thesis by presenting concepts relevant for the study. It starts with a review of human cognitive limitations together with information about coding, colour and feedback. Thereafter comes an introduction to basic HCI, interaction design and usability concepts and the chapter continues with a thorough introduction to usability testing. The last part concerns methodology theory.*

---

As stated above, the chapter starts with theories about the cognitive characteristics of humans. These theories are important to understand as possible limitations that must be taken into consideration when designing an interface. To understand the relation between the human characteristics and computers, theories about HCI and usability follow the cognitive psychology parts. Thereafter follows methodology theory. Since the main part of the thesis consist of the heuristical evaluation and the usability test, the theory segments about these parts have been given own sections outside the methodology chapter, to emphasize their importance.

#### **2.1 Cognitive psychology**

Eysenck and Keane (2000) claims that the world is getting more and more complicated and with that the understanding of it is getting more advanced. To be able to cope with the demands from the world we live in we must understand each other and our own inner mental spaces. Cognitive psychology deals with that sorts of issues, and are defined as a research area that involves different phenomena constituting the essentials of an individual's cognition. These are perception, learning, memory, language, emotion, concept formation and thinking. The cognitive psychology areas are despite its broadness unified in an approach called the information-processing approach. This approach is based on the similarity of the mind of the computer and has been the dominant paradigm within cognitive psychology for some decades (Eysenck and Keane, 2000).

### 2.1.1 Information processing

Information-processing psychology is, according to Artman (1999), the study of and tries to explain mental structures and processes by analyzing the process of how information is dealt with from the moment it is considered input to the human until a response is given. To be able to see it this way, the human must be considered an information-processing being and its mental models must be said to limit and transform the incoming data to process able and significant data. What constitutes significant data varies from individual to individual, but the important part is that the data later can be altered to become more meaningful and serve as a goal oriented response to the initial input. The information-processing approach concerns structures that can be learnt and created by interacting with the environment, rather than inherited skills (Artman, 1999).

Broadbent (1958) claims that a significant large part of cognition consists of different stages ordered in a sequential series. When stimuli is presented basic perceptual processes occur, followed by attentional processes that serves the purpose of moving some of the perceptual impressions to the short term memory. By rehearsal of that information some is left in the short term memory and other parts are transferred to the long term memory. The restraint of this approach is that it assumes that stimuli affect an unprepared and inactive human, when in reality processing often are biased by earlier experiences and expectations. Eysenck and Keane (2000) differentiate between bottom-up processing and top-down processing. Bottom-up processing, or as it is also called stimuli driven processing is distinguished by its direct effect when a stimulus is presented. Top-down processing or conceptually driven processing is affected by what the person brings into the situation in reference to earlier experience, influence of context and similar things. The sequential stage model deals first and foremost with bottom-up processing. During the 1970s it was heavily argued that almost all cognitive activity constitutes of both bottom-up and top-down processing occurring at the same time. At the end of that decade most of the active cognitive psychologists agreed that the information processing approach was the best way of studying human cognition (Eysenck and Keane, 2000).

The information processing approach is under continuous development as parts of it have emerged from the view that the human brain resembles a computer, and computers are under constant development. During the maturity process of cognitive psychology the similarity to computers has helped researchers to understand the mind, based on how computers work. Since the origin of parallel machines theorists have now returned to the notion that cognitive theory should be based on the brain's capacity of parallel processing (Eysenck and Keane, 2000).

### 2.1.2 Perception

Perception is what really starts the information processing, and one definition given is:

*“The term perception refers to means by which information acquired via the sense organs is transformed into experiences of objects, events, sounds, tastes, etc.”*

(Roth, 1996, p. 81)

Lundh, Montgomery and Waern (1992) describe perception as the way we interpret information that has been presented to our senses. Out of the five senses possessed by humans (sight, hearing, touch, smell and taste) sight processes about 80 per cent of our total information input (Dodwell, 1994). Information perceived by sight is said to be visually perceived and visual perception is a very complex process that deals with encoding and interpretation of stimuli given by the environment. Another aspect of the perceptual process is sensation which is the first contact between the outer world and the human sense organs. Perception is what we do with that collected information once it reaches the visual cortex; we sense the presence of stimuli but we perceive what it is (Eysenck and Keane, 2000).

### **2.1.3 Perceptual organisation**

The first step when analyzing a visual scene is to separate different objects from the background. This can be affected by the brightness, size, different contours or other cues that might lead the person to a certain opinion. In the next step, the Gestalt laws are used to organise the visual field. The Gestalt psychologists were a group of German researchers that carried a thesis about perceptual organisation saying that the whole is more than the sum of the parts. This thesis is supported by the Law of Pragnanz that constitutes that the final organisational process will be as regular, simple and symmetrical as the conditions allow. Meaning that the perceptual systems work in order to create a correctly interpreted perceptual model of the real world, including the essence of what perceived.

As stated above, the information perceived by the visual sense is organised according to the Gestalt laws, sorting them depending on their proximity, similarity, continuity and closure. The Law of Proximity explains that elements that are located closely together tend to be seen as a figure or a unit. The Law of Similarity says that objects that are alike often seem to be grouped together. Elements that seem to follow in the same direction is often seen as a figure according to the Law of Good Continuation and the Law of Closure explains that when a space is surrounded by a contour it is often perceived as a figure (Coren, Ward and Enns, 1999). These Gestalt laws must be taken into consideration when designing a computer interface. It is according to them that users perceive the display and extract the information they need. Proctor and van Zandt (1994) emphasize the importance to use these principles in the design process to highlight and make the necessary information become as visible as possible.

### **2.1.4 Attention**

The world we live in provides us with a constant stream of stimuli through our sensory preceptors. These stimuli feed us an enormous amount of information about our context, but with our limited possibility to process all that information a filtering procedure must be executed. We turn our attention to parts of that information and the rest falls into the periphery of our consciousness (Coren et al. 1999). This fact was known already in 1890 when William James spoke of attention:

*“Everyone knows what attention is. It is the taking possession of the mind, in clear and vivid form, of one out of what seem several simultaneously possible objects or trains of thought. Focalisation, concentration, of consciousness are of its essence.”*

(James, 1890, pp. 403 – 404)

The phenomena of attention can be classified depending on how many objects that are in focus at the same time, and are divided into focused attention and divided attention. Focused attention is when a human is presented with several different inputs but chooses to respond only to one of them, divided attention on the other hand is when experiencing several different stimuli and attending and responding to all of them.

### **Focused attention**

Eysenck and Keane (2000) draw the resemblance between focused attention and a spotlight, everything that falls into the attentional spotlight can be seen clearly but everything outside it is much harder to see. A shift in the attention is assumed to happen when the spotlight is moved, and it can be assumed that the spotlight is moved at a constant rate. Another view of this was presented by Eriksen and St. James (1986), who claim that attention is like a zoom lens that gives attention to a certain region of the visual field. This region can be changed in size, depending on the demands of the task. Studies show that subjects can, in order to avoid noise and disturbance, zoom in precisely at target stimuli when other stimuli are presented in an area close to the target stimuli.

### **Divided attention**

When people try to do two things at the same time it can cause problems. Depending on what two things tried to do it might go without problems, but the tasks might also disturb each other causing a breakdown in performance. This kind of breakdown is said to shed light on limitations of the human information-processing capabilities. Some theorists claim that the human attention has limits in doing several tasks at the same time, reflecting on the breakdowns suffered when trying to conduct multitasking. Others concentrate on the impressive fact that humans actually can perform two rather complex tasks at the same time without breakdowns. These scientists claim that as long as the two tasks use different processing abilities there will be no interference between them (Eysenck and Keane, 2000).

### **2.1.5 Pattern recognition**

Pattern recognition involves, in a very general definition, identification and categorisation of stimuli and matching that with earlier stored information in the memory. There are two different theories describing the phenomena of pattern recognition, the template theory and the feature theory. The template theory claims that there are miniature copies of all recognisable patterns we know stored in the long term memory, and based on how we perceive the stimuli of a pattern the closest match from the memory is used. Since visual stimuli that are supposed to match a certain

template can be altered in many different ways and therefore can be hard to recognise, there has been some improvements in the theory. It is claimed that the visual stimuli goes through a normalisation process before the search of a matching template in the memory is started. The normalisation could be helpful when searching for digits and letters, but cannot be expected to produce a correct matching template consistently (Eysenck and Keane, 2000).

The feature theory claim that a pattern consists of several features or attributes. The pattern recognition process in the theory starts with an identification of these features and they are later combined and compared with feature information stored in the memory. One advantage with this approach is that the visual stimuli can differ greatly in size and other details but will still be able to match in a certain pattern (Eysenck and Keane, 2000).

### 2.1.6 Mental models

To facilitate the use of a system it is important that the designer provides the user with a good conceptual model that enables the user to learn and use the system in an effective way. When users learn and generate knowledge of how to use the system and to a lesser degree understand how it works, it is said that the user have created a mental model of the system (Preece et al., 2002). A mental model is described as the way a user perceives a system and its underlying processes that help them use it. Good design is when the designer’s conceptual model matches the user’s mental model of a system, or if a new product is being developed when it makes sense to the user (Barnum, 2002). The designer and the user communicate through the system image, which is built on the user’s perception of the physical structure of the system. If the system cannot communicate a clear and consistent system image to the user he will end up with the wrong mental model. Therefore it is vital for the designer to start with a conceptual model that is learnable and usable, and make sure that the system can expose the right system image. If that happen the user will end up with a correct mental model of the system and will use it in a correct way (Norman, 1988).

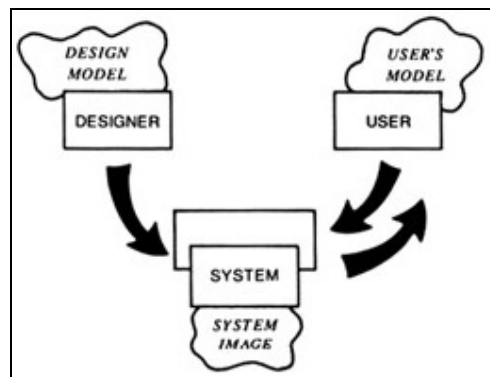


Figure 5. Norman’s figure of mental models (Norman, 1988, p.190).

The usage of erroneous mental models is more common than one would think. Preece et al. (2002) mention an example of people who starts pushing every button they can on their mouse and keyboard when the cursor freezes. But when asked what good they

think that will do, the explanations given are quite vague. Another example is when people starts hitting machines that behave in an unexpected ways. It can be said that mental models of interactive systems often are wrong, incomplete, easily confusable and based on wrong analogies. Not having the right mental models causes frustration and releases strange behaviour as mentioned above. To avoid this behaviour a clear and consistent system image must be provided, and this should include: “

- *useful feedback in response to user input*
- *easy-to-understand and intuitive ways of interacting with the system*

*In addition, it requires providing the right kind and level of information, in the form of:*

- *clear and easy-to-follow instructions*
- *appropriate online help and tutorials*
- *context-sensitive guidance for users, set at their level of experience, explaining how to proceed when they are not sure what to do at a given stage of a task.”*

(Preece et al., 2002, p. 94)

### **2.1.7 Feedback**

Feedback is the information sent from a system showing a user that the performed action had an effect on the system status. When a user performs a choice in a system, information should be sent to the user concerning that a change has happened and what effect that change has on the system. That is fundamental to understand what effects user actions have on a system (Norman, 1988). One sort of feedback that is crucial to include in software is error messages. Error messages must provide the user with information about what the problem is, what can be done to recover from it and how the user can avoid getting stuck with the same problem again. There are many different kinds of feedback, for instance audio, tactile, verbal, visual and various combinations of these. Deciding what combinations that should be used in a system is vital, and depends on the purpose of the software. Providing good feedback within an interface makes the interaction easier for the user (Preece et al., 2002).

### **2.1.8 Colour**

The ability to distinguish colour have two major values to humans, detection and discrimination. Detection is said to be the way we separate an object from the background and discrimination is how we detect small differences among objects (Eysenck and Keane, 2000). The appearance of a colour can vary in three dimensions; hue, which is the nameable aspect of colour (green, red etc.), brightness and saturation which responds to a colour's physical dimension of purity. The purest colour you could get would correspond to a monochromatic hue. Monochromatic comes from the Greek language, mono meaning one and chroma meaning colour (Coren et al. 1999).

The use of colours in computer software have a variety of reasons, to group similar things, facilitate the separation of dissimilar things, to show temporal differences or just to label things. It can also show relations between different parts of the interface or be used to draw attention to a certain part of the screen. Surveys show that even though colours in a user interface not always enhance the user performance it is likely to have an affect on the user acceptance to the system. Users think that they perform better in systems with colours, presupposed that the colours are used in the right way (Meier, 1988).

People perceive colour in different ways and observing a coloured object does not just provide additional information about a stimuli, it also has emotional connections. Certain colours are known as warm (yellow) and others cold (blue), sensations that really must be experienced by the sense touch. Colours are attached to different experiences depending on who is observing them, and it is said that colour is a psychological experience as much as a visual one. This constitutes that a design choice regarding colours not only affects how the user perceives the visual part of an interface but also can affect how the user feels (Coren et al. 1999).

When working with colours on computer screens it is important to remember that there are two highly powerful information-processing units at work (the human and the computer), but all communication between them must pass through the display which chokes off all fast, complex and precise communication. This is why it is important to avoid noise in designing an interface, by trying to use the good properties in colours that improve the information resolution (Tufte, 1990).

### **2.1.9 Coding**

Information coding in computer interfaces have been subject for many research studies, showing that a number of human capabilities must be taken into consideration during the encoding of the information. Based on these findings guidelines have been developed to aid designers in what can be a critical point of the software design. When using a single graphical code Hertzler and Nowell (1998) suggest the following ranking, based on reaction time for visual search tasks: colour, size, brightness or alphanumeric, and shape. The finding that colour is the most effective coding device has been confirmed by other studies, though these studies claim that next to colour, shape followed by letters or digits are most effective. Despite their ranking of shape, coding using shapes that are alike to a large extent actually increases search times instead of decreasing it (Hertzler and Nowell, 1998).

Using colour coding can solve many problems, but it might also cause a number of problems if the coding is not correctly done. Meier (1988) mentions the example of using green as a notion of acceptance, yellow as representative for caution and red as a sign for stop or error as a known example that would be fatal to misuse. One problem when designing interfaces is that the choice of colours are often made without considering their physical or psychological aspects, and without considering colour user guidelines and design suggestions. Colours used improperly might even do worse damage to the interface than what no colours would have done. It can be confusing to the users, cause eyestrain, and suggest that a system is working when it is really not or any other colour related issue. A system that is correctly labelled with

text might lose that usability aspect if the colour coding is inconsistent with the rest of the interface (Meier, 1988).

Redundant information coding is when two graphical devices, such as colour and shape are used to code the same information. Studies have shown that using redundant coding facilitates an even faster search than a single coded piece of information would have, especially when combining the mentioned couple, shape and colour. These benefits increase even more when the interface is very complex or cluttered (Hertzler and Nowell, 1998).

### **2.2 HCI – Human Computer Interaction**

Human Computer Interaction can be defined in numerous different ways, SIGCHI (Special Interest Group of Computer Human Interaction) uses the following, concluding that HCI is:

*“...concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them”*

(ACM SIGCHI, 1992, p.5)

Winograd (1997) describes HCI as a field that by necessity consists of interdisciplinary concerns, since its core is interaction that involves people and machines, virtual worlds and computer networks, and a broad array of objects and behaviours. Another definition that also is used claims that HCI is:

*“the study of the relationships which exist between human users and the computer systems they use in the performance of their various tasks”*

(Faulkner, 1998, p.1)

The term HCI replaces the old MMI abbreviation, meaning Man Machine Interface although Faulkner (1998) thinks that a use MMI sometimes would be preferable since it implies a breadth of study which can be useful in the development of human-computer systems. The task of HCI is to design for people, the tasks they perform and the environments the tasks are performed in. If the HCI process is to be effective it is important to consider all these elements in the development of new human computer systems.

This view is shared by Gulliksen and Göransson (2002) who also stress the importance of taking the users' background, abilities, limitations and working environments into consideration when developing an interactive system.

### **2.3 Interaction Design**

The term Interaction design was originally formulated because the HCI definition was not broad enough. Interaction design contains, except for HCI themes, issues that have been considered beyond HCI. These are according to Preece, Sharp and Rogers (2002) for instance, psychology, web design, computer science, information systems, marketing, entertainment and business. Their exact definition of interaction design is:

*“designing interactive products to support people in their everyday and working lives”.*

(Preece, et al., 2002, p.v).

Winograd (1997) claims that interaction design as a profession comes from this clash between different kinds of research disciplines, and that it has its own distinct set of concerns and methods despite drawing many from older disciplines. Interaction design is not a subfield of computer science but an own field of research with foundations for designing interaction with computer based systems found in the HCI elements of graphic design and information design.

Norlin and Sjöberg (2002) claim that there is a resistance to hire interaction designers since they are thought to add too much cost to a project. To give an example they use a business developer’s statement saying that customers never go for the entire deal of interaction design, but that they only want parts of it, as they can do when only buying an interface. But that statement has its base in the wrong definition of interaction design, according to Norlin and Sjöberg (2002). Interaction design is the expertise that adds higher quality to a product, that communicates lifestyle status, that excludes confusion, that adds flexibility and that makes a product be desirable. It is believed that a variety of reasons is to blame for the understanding that interaction design is just an expensive add-on to a project, for instance the failure for people to understand that a good product are both functional and pleasurable. This means that it works properly in its deliberate environment and it is desirable for people; it gives them a great feeling to own. Apart from that, it can be said that interaction design does not fit very well in to the existing development cycle for products since it clearly is a very iterative process; to optimise the usage of interaction design the entire process should be altered.

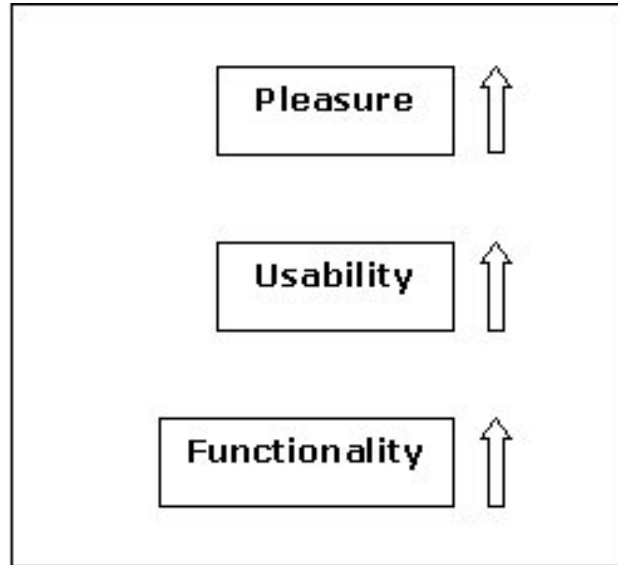
### **2.4 Usability**

According to Preece et al. (2002) the usability of a product depends on the user’s perception of the product. This perception is built on if the user feels he can interact with the product easily, learn and relearn it easily, how intuitive the product is and how much the user appreciates the usefulness of the product. Usefulness is defined by Preece et al. (2002) in terms of the user’s need for the product in context of the user’s goals. This means that usability applies to a certain user, performing a certain task in a certain context. That view of usability is also supported by ISO, the International Organization for Standardization that defines usability:

*“The extent to which a product can be used by specified users to achieve specified goals in a specific context of use with effectiveness, efficiency, and satisfaction”.*

(ISO 9241:11, 1998)

Jordan (2002) claims that except from usability, people also look for pleasure in products. In order to create a pleasurable product, the first step is to make it functional. Jordan (2002) describes a hierarchy of consumer needs (Figure 4) and states the different levels are depending on each other.



**Figure 6.** A hierarchy of consumer needs (Free from Jordan, 2002, p. 6)

The first level is evidently functionality; the basic requirement for any product is that it must be functional. If it does not manage the operations it is said to do, people will never accept it and it will cause dissatisfaction. A dysfunctional product will soon disappear from any market it is launched in.

The second level, according to Jordan (2002), is usability. People are getting used to functional products and start to discover issues in them that could have been solved better. They want usable products, which is dependent on functionality. A functional product is a prerequisite of usability, but it does not guarantee usability.

The third level is pleasurable products. When people have been using usable products for a certain time, they want something more. Products that not merely serve as tools, but can be related to as living objects and bring the users not only functional benefits but also emotional. As with usability, pleasurability is dependent of the level below. A usable product does not have to invoke pleasure emotions in its users but a product that has no usability is unlikely to be pleasurable. This means that usability is a key step in the striving towards pleasurable products, and a first aiming point for products that are in a functionality state (Jordan, 2002).

To achieve the usability level it is important to reflect on how decisions about product design are taken. Faulkner (1998) claims that when designing an interface all decisions about usability should be taken by a designer. In that way all decisions related to the usability of the system have been taken consciously and explicitly, with the designer able to motivate and explain them all. Unfortunately not all decisions are made like this, and some might be taken unconsciously without the developer's intention to make a design decision. One way of avoiding such mistakes is to clearly state the goal of the system and have a close collaboration within the design team (Faulkner, 1998). By stating those goals, the product development process has good possibilities to produce a usable product which might be popular in the market. As said by Bevan, Kirakowski and Maissel (1991), usability is the way users measure the quality of a software product.

### ***2.5 User-adapted interfaces***

A user adapted interface is an interface that adapts to each end user by knowing key features about them and thus applying the appropriate interaction. This known attributes are based on pre-recorded information about a specific user, and can by that way facilitate interaction adapted to that exact user. Since there are numerous of different attributes that a user can exhibit it will require quite large differences in the dialogue styles that are presented by the interface. It can be said that to a user-adapted interface there is no average user, only a large number of possible combinations of attributes which gives the conclusion that not all combinations of attributes can have a design of their own. Instead one or a few different basic designs with a number of alterations are used (Akoumianakis, Paramythis, Savidis and Stephanidis, 1997).

### ***2.6 Usability testing - definition***

What constitutes a usability test can differ greatly depending on the person who does the interpretation of the expression. There are examples of people who use the term when referring to any technique used to evaluate a system or a product. Others distinguish between techniques that involve actual users and techniques involving experts such as heuristic evaluations and cognitive walkthroughs (Barnum, 2002). Within this study the expert evaluations are seen as a part of the process of making a system more usable but the term usability tests will refer to tests carried out with authentic users.

### ***2.7 Heuristic evaluation***

A heuristic evaluation is an expert method used to find usability problems within a user interface. The method can be used in an iterative design process or as an evaluation method of complete software. It is carried out by letting a group, preferably three to five, of evaluators examine and compare the user interface of interest to a set of usability principles called heuristics (Nielsen, 1990).

The underlying principle of heuristic evaluation is, according to Barnum (2002) that the rules of the evaluation are logical and intuitive for any expert to use. This is easily achieved by simply setting a set of heuristics and letting each expert evaluator start their work from that set. This fact facilitates the evaluation by giving the experts a possibility to focus on the same categories for the purpose of cataloguing the results. These heuristics can also be seen as guidelines for designing a user interface.

Nielsen (1992) claims that the optimal way to use heuristic evaluation is to use double specialist during the process. Double specialists are usability experts who also possess experience about the interface being evaluated. Nielsen admits though, that double specialists not always are available and that regular usability specialists also find a high percentage of usability problems in an interface.

Both major and minor usability problems are often found within the heuristic evaluation results. Many minor problems that are found in heuristic evaluations might not have been found in actual user tests. The severity of the different problems can be graded after the evaluation and ones that are labelled as major problems often includes such issues that have the potential to confuse the user and cause them to use the system erroneously. It is then up to the corporation's usability engineers to prioritize and make sure that all major problems are solved first, although that does not exclude the solving of the minor problems at the same time (Nielsen, 1992).

### ***2.8 Usability test***

According to Wichansky (2000) there seems to be no substitute for actual user data when looking for answers if a product is usable. When the question of usability arises, people want to hear if real users can manage the system and what they think of it, not a list of usability problems suggested by a usability expert. To collect those data, an empirical study involving users would be the best suggestion. Barnum (2002) claims that usability tests are most effective as a diagnostic tool to learn what problems users have with a product. She means that qualitative data from four or five tests with users can clarify many problem areas within a product. Dicks (2002) agrees that a limited testing with few participants will help to do major improvements to products but it cannot verify with absolute certainty that a product is usable.

Dumas and Redish (1999) claim that a usability test must cover five characteristics; the goal must be to improve the usability of a product, the test participants represent real users, the participants perform real tasks during the test, that the test leaders observe and collect data during the tests, and that the test leaders analyze the collected data and recommend improvements to the product.

#### **2.8.1 Test plan**

The test plan should serve as a blueprint for the entire test, including information about test team members, elements of the test and details concerning the test. Depending on the expectations of the test report and its intended readers the test plan might be of different formats or formality, but it is always good to have all the decisions in writing. This is also important during the test process, making sure that

the test team members can locate the goals and aims of the test whenever they need a reminder of what was agreed on at an early stage of the process. The plan also describes what resources that are needed to conduct the test and serves as a running planning document for the entire process. When the test plan is written it is decided that a test will be conducted, almost all other things in the plan are possible to alter or reschedule so it is important that the team updates and keeps informed of its contents (Barnum, 2002).

### 2.8.2 Test participants

Before any test can be performed the test participants must be chosen. This is a very important part of the usability test process since a bad recruitment can affect the entire test result. The test participants must represent a possible final user of the system and therefore basic characteristics demands should be fulfilled. These characteristics should be written down and consulted when choosing all participants in order to secure valid testing conditions. Since people's motivation to participate varies, along with their commitment to actually show up for the testing session, backup participants should always be arranged. This ensures that the test can be carried out in time despite any sudden changes in the agenda (Kantner, 1994). When it comes to choosing how many test participants that should be included in the study there are many different opinions. A starting point can be Rubin's statement:

*"...it is better to test than not to test"*

*(Rubin, 1994, p. 27)*

Knowing that, it is still argued of how many test that must be performed to find a sufficient large percentage of the usability problems in a test object. Dumas and Redish (1999) claim that a typical usability test includes five to twelve users, but there are often specific constraints that have to be taken into consideration and those might therefore lower the test participant number. Holleran (1991) agrees that no test group should include fewer than five participants and that a larger number is desired. The opinion that five users would be enough to find originally came from Nielsen (1993) saying that three to five users could find as many as approximately 70 per cent of all usability problems in a system. This claim has later been criticised and investigations show that a groups of five users can find a percentage of problems as low as 55 per cent depending on the participants were recruited and grouped. A slight increase to ten participants turned the minimum percentage of faults found to 82 per cent with a mean of 95 per cent. Although those numbers clearly show that ten participants found more usability problems that five, it all depends on which users that are recruited. Faulkner (2003) claims that the most important and complex issue when dealing with usability test recruiting, is to find correct and representative users.

When dealing with people, as normally with a usability test, it is important to make the test participants feel comfortable. It is the test leader who should make sure that such is the case and stress that it is not the participant who are being tested but the system. People tend to feel inadequate or stupid when making errors or not understanding the system, especially when knowing they are being watched or recorded. Therefore it is vital to ease the participants into the test, first by properly

welcoming them to the test and offer the available refreshments followed by a thorough introduction to the test. It is also recommended that the first task in the test itself is a very easy one, basically guaranteeing a good and self confidence enhancing start to the test session (Nielsen, 1993).

### **2.8.3 Constructing test**

In a usability test it is common to try to make the test participants accomplish typical system tasks. One way of doing this is to use a scenario based test involving these typical tasks in the scenarios. Tasks can be decided in different ways but frequently used criteria involve: first impressions, first tasks, regularly performed tasks, critical tasks and specific problem areas (as identified by heuristic evaluation). Barnum (2002) claims that it is rather easy to come up with tasks that would be useful to test, and that the hard part is to convert them into working scenarios. The two major issues that should be given some extra thought are; focusing on the process - not the steps to get there, and avoid using language of the system – it is important that users understand what they need to do. The test activities must be based on the objectives for the test, enabling the collection of correct data.

The test tasks need to be small enough to make sure the participant can complete them within the time scope of the test, but not so small that they feel trivial. Another aspect to think of is to make the tasks business related, and avoid humorous or non-serious tasks (unless that is the purpose of the product) since there is no guarantee that all participants have the same sort of humour and it gives the entire test a less serious label. As stated above it is suggested to start the test with a rather easy task to make the participant feel at ease, but it is also important that the test end with something that the participant can handle so he will leave the test with a feeling of having accomplished something (Nielsen, 1993).

### **2.8.4 Walkthrough and Pilot test**

Nielsen (1993) claims that no usability test should be conducted without first trying out the test procedure on a few test subjects in so called pilot tests. This view is supported by both Barnum (2002) and Kantner (1994) but these researchers also favour a reading of the test prior to doing the pilot test. This reading is called walkthrough, Barnum, or dry run session, Kantner. The walkthrough is the first chance for the test team to practise the test procedure and to discover problems with the tasks or how the test is planned. By instant feedback the test team can see which parts of the tests that might have to be redesigned, and since a walkthrough should be planned enough in advance to the real tests there should be no problem in finding the time to apply these changes. This pre test session is also good practise for the test leader who gets experience in talking from the script, modifying it as needed. One important aspect in the walkthrough is who the test person is, he should be a very tolerant person since he might have to put up with computer crashes, mistakes by the test leader, confusion or other errors in the test process. After the walkthrough is conducted it is recommended to have a debriefing session within the test team to discuss what changes that have to be done and what to think of when it is time to do the pilot test (Barnum, 2002).

The pilot test is the usability tests' version of theatre's full dress rehearsal, a final check of the test procedure and the estimated time to perform the test. The whole test process should be conducted including pre test instructions and post test questionnaires. In this stage information about too hard or too easy test tasks is reachable and there might still be some changes needed to the test. The pilot test might also be used to refine or clarify what are being measured in the tests, for instance what the characteristics of a user error are (Nielsen, 1993). The recruit for the pilot test should be from the group of possible test participants to make the pilot really useful. If everything goes smoothly, the results from the pilot can just be added to the final results with one extra participant. If the pilot test uncovers new problems that were not seen in the walkthrough there should still be time to correct these before the real tests begin. Collected results from a pilot test can also be used as training in analysing data, facilitating that process when the real results are gathered (Barnum, 2002).

### **2.8.5 How to conduct the test**

When the test participant arrives at the test location is important to make him feel welcome and as the resource of improvement he really is. Therefore it is recommended that someone from the test crew sets aside time to greet the participant and make him feel at ease, this could be done by offering refreshments and a place to wait until the test crew is ready. While waiting for the test crew a short pre-test briefing session can be conducted, informing the participant what is going to happen during the test and letting him fill in any questionnaire needed. Information that also might be of interest at this point are a short description of the system being evaluated, an explanation of the think-aloud technique and most importantly a clear emphasize on that it is the product being evaluated not the user (Barnum, 2002).

According to Nielsen (1993) there might be a need to train or introduce the participants to the system before the actual test begins in order to get any data from the test at all. If the system is totally unknown to the participants it might produce flawed data that causes test team to draw the wrong conclusions when they analyse it.

During the real test it is important that the test leader tries to keep objective and that he restricts his interaction with the user. He should not make any comments on how the test participant is doing and it is recommended that he does not help the user, even if the user are getting into serious difficulties. This comes with one exception, if the user is clearly stuck and is starting to get annoyed it is suggested that the test leader helps him through, also if a user is starting to have problems with a task where earlier tests have been known to collapse he should be able to help him (Nielsen, 1993).

Kantner (1994) argues that it is very important that each participant experience is consistent to the next to provide for a common ground for the collected data. One way of providing for good consistency in the tests is to use a test script throughout the whole test process.

### **2.8.6 Think-aloud protocols**

Think-aloud technique is a way of understanding what a testing subject thinks when conducting the test. It requires people to verbalize their thoughts during an entire test, including what they are doing, what they thought of doing and how they react to anything the software does. A problem with the technique though, is the occurrences of silent moments. One way of trying to avoid these would be if the test leader kindly could remind the test participant to think out loud, but this is not recommended since it would disturb the participant too much according to Barnum (2002).

Holleran (1991) says that it is not normal behaviour for users to think out loud when doing a task by themselves, and it might therefore be distracting when using an unfamiliar piece of software. It is said that using the think aloud technique does not alter the sequence of cognitive processes, but studies (Ericsson and Simon, 1980, 1984; Russo, Johnson and Stevens, 1989) show that verbalization during performance of tasks may have effects on the time it takes completing the task, the way the task is solved, success, attention to details as well as information retention. Therefore Ericsson and Simon (1980, 1984) have suggested a number of aspects that should be concerned, for instance the time of verbalization. They claim that verbal reports of cognitive activity have validity if the subject can access information from his short-term memory, which means that he should think out aloud directly as he concludes a task. Other factors that also influence the quality of the collected data are motivation, practice and attention. Due to these weaknesses in the data collecting method it is important that any test leader does not merely rely on think aloud protocols, but judge them together with other objective data.

### **2.8.7 Structuring the results**

When the tests have been completed the researcher has collected a great amount of test data. These data need to be organised in some way to ease the analysis process and facilitate the result report. One way of categorising the results is in a top-down approach which most often starts with a set of categories for potential findings from a usability test. These categories are frequently constituted by a set of heuristics; whenever a heuristic is breached the finding is sorted in that category. The set of heuristics can be the same as used when conducting an expert evaluation of a system, but there are others more suitable for categorising results from usability tests. One advantage of using a top-down approach is that when a test team consists of only one or two people, the heuristics help the researchers to be alert for other findings than just the expected issues. It has also proven to be one of the most consistent analysis methods (Barnum, 2002).

## **2.9 Methodology Theory**

The following sub-chapters of the study introduce methodology theory in general and some aspects of it that are highly topical for this study.

### **2.9.1 Literature studies**

One of the most important steps when conducting a study, according to Backman (1998), is the literature review. It helps the researcher to see how other similar studies have been performed, indicating suggested tried ways but also known weaknesses and knowledge gaps in the area of the study. The review aids the researcher in planning how to conduct his study by providing information about other investigator's dispositions and analysis methods. Backman (1998) claims that in many cases the possible success of a study depends on a well performed literature review.

### **2.9.2. Approaches of research**

When conducting a study it is common to differentiate between two separate approaches, quantitative and qualitative studies. Qualitative studies are primarily used to understand something, focus being on collecting data to achieve a greater understanding of the problem at hand. They also give an overall picture that facilitates the possibility for understanding relations within the scope of a study. Qualitative studies aim at finding out individual's subjective opinions about a certain topic (Holme and Solvang, 1996/1997). Hartman (1988) defines qualitative studies by stating that they can be characterized by the attempt to reach a deeper knowledge of how individuals or groups see themselves and their relationship with the context.

Quantitative studies on the other hand are used to make statistical generalizations and with the result at hand the researcher can make relative certain statements about conditions within the subject group. A quantitative study is therefore obviously concerned with measurable amounts, how much there is of something. (Holme and Solvang, 1996/1997). According to Hartman (1988) the quantitative studies can be divided into three phases: planning, collecting and analyzing. Planning includes the formulating of a hypothesis and general planning of the study, collecting involves the actual accumulation of data, and the last phase involves analyzing and comparison of data to the hypothesis. Hartman (1988) claims that quantitative studies can be characterized by the existence of a numerical relation between two or more measurable properties.

### **2.9.3 Interviews**

An interview is a conversation based on a certain purpose and containing a certain structure. There are many different kinds of interviews, and among these is the research interview. Kvale (1996/1997) defines the research interview as an interview

which purpose is to receive descriptions of the interviewee's world with the intentions to interpret the meaning of the described phenomena.

Barnum (2002) claims that participants of interviews are more likely to go into deeper details about opinions and attitudes in their answers, than they would if they had answered the same question in writing. Preece et al. (2002) agree, but also state that the choice of data collection method should depend on what the goals with the study are. An informal, unstructured qualitative research interview is suitable if the researcher is interested in user design ideas. According to Kvale (1996/1997) there are many different kinds of interviews, structured, semi-structured and unstructured being the ones most used. Qualitative studies often include semi-structured or unstructured interviews. The semi-structured interview is based on a few predetermined questions (interview guide) that the interviewer bases the conversation on, but the interviewee should be allowed to speak freely if wanting to. An unstructured interview basically has only one pre-decided question and thereafter the interviewer improvises depending on the answers from the participant.

Preece et al. (2002) state that one benefit of unstructured interviews is that they gather rich data, meaning that the interviewee can mention data that the interviewer did not even think of and thus broaden the exploration possibilities within the study. The disadvantage of this is the amount of data generated; the larger amount of data the longer time to analyze. Another negative aspect is that by choosing an unstructured interview, the researcher makes it hard to regenerate the process since every interview takes its own format. Semi-structured interviews in contrast are intended to be regenerated and repeated, which aids the reliability of a study (Preece et al., 2002).

In order to ease the remembrance of what have been said in an interview Kvale (1996/1997) suggests a few different techniques. The most common way of register an interview is to record it on tape, letting the interviewer focus mainly on his questions. Although this technique helps the researcher to remember what have been said during the interview it presents a decontextualized image of the situation since no body language or other visual elements can be detected. In order to see those aspects, the interview must have been recorded with a video camera, documenting all the visual and audible data. Other techniques that are used are note taking during the interview, often combined with great trust of the researcher's memory of the situation. One problem with that technique is that the researcher easily can forget details about the situation and it is recommended to start working with the data rather quickly following the interview (Kvale, 1996/1997).

### **2.9.4 Questionnaire**

One way of getting feedback about user's opinions about a system is to conduct a survey, often consisting of a questionnaire. It is important to make sure that the questions in the questionnaire are unbiased and really encourage the users to answer their own opinion and not what they think the test leaders want to hear. By using the Likert scale as an alternative to yes or no answers no user is forced to choose one or the other option; instead they can compare their opinion in to five alternatives and chose the most appropriate. The Likert scale consists of five different answer alternatives that must be put in relation to the question: strongly agree, partly agree,

neutral, partly disagree and strongly disagree. When other scales that include more alternatives (7-point scales) are used, the participants have a tendency to avoid the extreme alternatives on both ends which bias the result of the survey. Other questionnaires that are recommended to avoid are such that includes too many open questions combined with a too large space for participant answers. If open questions are used in a survey it is important to limit the number of them since they take time to answer for the participant as well as to analyze for the test crew (Barnum, 2002).

Post test questionnaires are very much alike to normal surveys, with the difference that they are conducted in connection to a usability test session and have the purpose to collect feedback on a recently used system. It is recommended to use the Likert scale also in post test questionnaires, making a statement and ask the participant to choose to what level he agrees. Despite the fact that the participant has an own opinion of the tested system after conducting the usability test, it is important to formulate the questions in an unbiased way in order to collect what the user really thinks of the system and not what he thinks the test leaders want to hear (Barnum, 2002).

### **2.9.5 Analysis**

When the observational phase of a study is completed the collected data must be organised and analyzed. The purpose with organising the data is to give it a structure which facilitates overview and interpretation of the overall findings. By analysing the outcome of the observational phase and putting those results in relation to the research question conclusions about the interface can be drawn. The interpretation of data is very complicated and demand great concentration of the researcher. It must be secured that all steps taken during the research process corresponds correctly to the research question, making sure that interpreting the results give the right answer to the right question. Since interpretation and analysis is subjective, it cannot be guaranteed that different researchers would reach the same results despite working with the same material (Backman, 1998).

### **2.9.6 Validity and Reliability**

When conducting a study the terms validity and reliability are very important. Validity is described by the question; to what extent do our observations reflect the phenomena or variables that interest us (Pervin, 1984)? Another way of phrasing that would be in the words of Kerlinger:

*“Are you measuring what you think you are measuring?”*

(Kerlinger, 1979, p. 138)

Malterud (1996/1998) claims that validation is a vital aid for the researcher to critically reflect on his study during the research process. The study must contain data that has been thought through, validated by reflecting if the collected data is valid and if so under which circumstances. Concerning usability testing, Holleran (1991) claims that it is just as important that the test participants are representatives for the potential

## THEORETICAL RESEARCH FRAMEWORK

---

user group as it is that the test tasks are representative for the kinds of tasks that the software are designed to facilitate.

Reliability is the term that is used when discussing the trustworthiness of a research study. In order to be able to claim that the results of a study are reliable the researcher must have taken certain aspects into consideration when conducting it. This aspects concern the collecting of data and how well and systematic the analysis of it is made. The researcher must also constantly review the study to be able to exclude elements that might introduce faults or erratic data into the research. By this constant awareness of possible errors within the data occasional flaws can be detected and removed early within the process of the study (Malterud, 1996/1998). Another aspect of reliability is that a collection of measured data on one occasion must match the same measurement on another occasion or by another researcher. This can be achieved by asking the same questions, or variants of same questions to measure underlying variables (Holleran, 1991).

## CHAPTER 3

### THE EMPIRICAL STUDY

---

*This chapter describes how the empirical study was conducted and which results that was obtained. It explains how the three different phases of the study was performed, the literature study, the feasibility study and the usability test. The first part of the chapter describes how each phase was designed and the second part of the chapter describes the overall findings of each phase. These findings will be discussed in chapter four in relation to relevant theory presented in chapter two.*

---

The empirical study was divided into three phases: the literature study, the feasibility study and the usability test. These studies were conducted in mentioned order and did all contribute to the overall findings that constituted the base of the recommendations for design changes in SpaceLab.

#### **3.1 Literature study**

Literature has been sought both online and physically at location at Chalmers' library at Lindholmen in Gothenburg. Databases used for search have been: ACM digital library, IEEE Publications, CiteSeer and Google Scholar. Within these databases searches have been made using search criteria such as: "usability", "usability test", "heuristics", "heuristic evaluation", "cognitive psychology", "perception", "attention", "pattern recognition", "colour", "colour coding", "information encoding", "think aloud", "HCI", "interaction design", "interviews", "questionnaire", "validity", "reliability", "scenario based testing" and "user-adapted interfaces". In those cases applicable, the search criteria have been used both in Swedish and in English. The information found in these searches was used in the understanding of the theoretical research framework relating to the thesis focus. It was also useful during the work process, providing suggestions on how perform parts of the studies and it gave some good proposals on how to interpret gathered data from the different test phases.

## **3.2 Feasibility study**

The feasibility study was divided into two parts. First an interview with a SpatialAce support worker was performed and later four heuristical evaluations of SpaceLab were carried out. The result from the feasibility study partly served as a ground for the development of the tasks for the actual usability test and partly as a collection of evidence of usability issues to be dealt with in the analysis phase.

### **3.2.1 Interview with SpatialAce support employee**

During the feasibility study a semi-structured interview with a SpatialAce support worker was conducted. The purpose of the interview was to learn what areas of SpatialAce the real users had found problems in and how it was solved through help from the Carmenta supplied support. In order to generate as much usable data as possible, the interviewee was informed about the interview a week and a half in advance, giving her a chance to recap on the concerned support matters. At Carmenta all support topics are logged into one database which makes it easy for any authorized person to see all matters concerning a certain flaw. This is the reason why I only interviewed one person, along with the fact that the result of the interview only would serve as background information during the rest of the study.

The interview was conducted at the workplace of the support worker, a woman who had worked with support matters at Carmenta for number of years. An interview guide containing the most important subjects (as thought of by the author prior to the interview) was used, but it did not serve as a script for the conversation. The answers to the questions were written down as well as other important data discussed during the interview session. These data were written down and summarized directly following the interview, as suggested by Kvale (1996/1997), helping the interviewer to remember all details.

### **3.2.2 Heuristic Evaluation**

As part of the feasibility study a heuristic evaluation of SpaceLab was conducted. Heuristic evaluations are said to be discount methods as they are cheap and effective to use. It can also be completed without involving too many external users (Barnum, 2002). That fact suited this study well, since the results from the evaluation would not only point out flaws within thy system but also serve as useful information when constructing the usability test. Therefore, few and effective evaluations at an early stage was critical for the continuous research process, making heuristical evaluations a good choice.

The evaluation was carried out by four different interaction designers, at a location chosen by each evaluator. All evaluations were performed by using the same set of heuristics (see figure 6). The interaction designers all had experience of heuristic evaluations and were not allowed to communicate their views of the system until after the evaluation was ended. Since they shared Swedish as mother tongue all instructions about the test were given in Swedish, but the scenarios included in the tests were, due

to the language of the system, described in English. One of these scenario tasks that were used in the evaluation was actually taken from the SpaceLab help section to give a hint of how hard the software's first example would be to perform. The evaluators were given an hour to get to know the system before the actual test started, as suggested by Nielsen (1990). They were also asked to make notes about the usability problems they found during the evaluation, including a reference to the appropriate heuristic. The set of heuristics were available on a piece of paper located next to the evaluators during the entire test.

When the evaluation phase had ended, this was chosen by each evaluator by stating that no more usability flaws could be detected, the test leader performed a short debriefing session. This session included a few questions about the evaluator's impressions of the system and the answers were noted together with the evaluators list of usability flaws. In the debriefing session was also a discussion about the severity of the usability problems, comparing what had been found and giving each problem a label of major or minor problem. These labels were based on Nielsen's (1992) suggestions that a major usability problem was one that have serious potential for confusing the user or causing him to use the system in a wrongly way. Minor problems on the other hand were such that slowed down the user and caused unnecessarily inconvenience. Nielsen and Molich (1990) claim that when deciding on if a usability problem is major or minor it is a subjective judgment based on just one or a few evaluators opinion, which might not agree with how a user would perceive it. The set of heuristics used to evaluate SpaceLab were these, developed by Nielsen and Molich (1990), containing ten usability principles.

<p><b>Visibility of system status</b> The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.</p> <p><b>Match between system and the real world</b> The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.</p> <p><b>User control and freedom</b> Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.</p> <p><b>Consistency and standards</b> Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.</p> <p><b>Error prevention</b> Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.</p> <p><b>Recognition rather than recall</b> Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.</p> <p><b>Flexibility and efficiency of use</b> Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.</p> <p><b>Aesthetic and minimalist design</b> Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.</p> <p><b>Help users recognize, diagnose, and recover from errors</b> Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.</p> <p><b>Help and documentation</b> Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.</p>
---

Figure 5. Heuristics for expert evaluations (free from Nielsen and Molich, 1990, p. 249).

### 3.3 Usability Test

The usability test was part three of the entire empirical study, but constituted the larger part of the thesis work. By following recommendations from different literature, such as Barnum (2002), Nielsen (1993) and Ebling and John (2000) the usability test process was thoroughly worked through and the different steps in that procedure are presented below. To facilitate for the test participants all documents they received or signed, such as the consent form, the post test questionnaire and the written test tasks were given in Swedish. These forms can all be seen in the appendices.

#### 3.3.1 Test plan

A test plan was constructed at an early stage in the research process to have a written reference to consult when anything became unclear. The decisions that constituted the base for the test plan were naturally changed if necessary. The test plan was

constructed with Nielsen's (1993) and Barnum's (2002) recommendations in mind and covered the basic areas of information needed. It was kept visible to all test members during the entire test process to avoid any mistakes or misunderstandings. The test plan can be found in appendix 1.

### **3.3.2 Choosing test participants**

The first user characteristic that were sought when searching for test participants was the ability to handle a computer. Only users that worked with computers at a daily basis were considered as possible test participants, given that it was computer software being evaluated. Since Carmenta wanted to investigate how new users perceived and interacted with SpaceLab, the second most important user characteristic was that they could have no experience from working with SpaceLab whatsoever. This means that all usability problems found in the tests are considered problems that new users could experience. Another characteristic that had to be viewed in the selection process was the possible participants programming experience. Apart from finding usability problems in SpaceLab Carmenta also wanted to see if there was any difference in how users with programming background greeted the software compared to users without programming background. Individuals with programming background were therefore defined as: persons who have taken several academic courses in programming (20 > academic points) or deals with programming at a daily basis. Test participants were chosen to fit five users with programming background and five without. In order to make sure that the participant fitted into these groups, a control question of programming experience was brought into the post test questionnaire. Considering age or occupation no preferred option existed but the outcome of the participant selection showed that students were one group that could take part in the studies since they could often decide their own schedule. Therefore students constitute the larger part of the test participant group. Except the participants who did actually perform the test, two backup users were contacted and kept alerted if there would have been a no show, along with the recommendations of Kantner (1994).

### **3.3.3 Constructing the test**

The scenario based tasks in the usability test were created with a starting point in the result of the feasibility study. Interview and heuristic evaluation results showed where major flaws could be found and along with typical tasks for SpaceLab these constituted the base for constructing the scenarios. As Barnum (2002) suggests both typical tasks as well as tasks related to certain problem areas (as found in the expert evaluation) were included in the test. Also a scenario relating to the user's first impression about the software could be included since the test participants had no experience of working with the system. The test tasks were after the test construction judged by the author and given an approximate time to accomplish, and these estimated times were later checked and revised during the walkthrough and pilot test. Conducting the walkthrough and the pilot test were seen as a revision of the test tasks and a possibility for the test team to practise their parts for the actual usability test later.

Measurement criteria and goals for each task were decided after observing Barnum's (2002) example of criteria for similar cases. These examples concerned installation and first impression of a software tool, so goals and measurements were closely studied and then altered to fit into the SpaceLab usability test procedure.

Following the walkthrough a number of changes to the test were made, these changes applied to areas where the walkthrough participant had experienced problems in understanding the task or where the test leader had discovered a better way of explaining the task. Changes that were made were for instance rephrasing of certain task description, some additions to the system introduction, a better explanation of where to find an example within the help section and a correction of an erroneous named operator.

These changes made were later tested in the pilot test, which was carried out as a real test in order to practice the test procedure and see if the test itself was correct. Most parts of the tests worked as hoped for, but one task had to be reformulated to enhance participant understanding. During the pilot test there were moments when the test leader had to help the participant to understand certain things, but that was not due to the test design but merely an evidence of a usability problem. As the participant of the pilot test fitted to the target group of the test, the results from the pilot test could also be used as a result in the study as well as any other test participants. This option was used, as Barnum (2002) recommended, since it created a larger foundation of usability findings to base the design recommendations on.

### **3.3.4 Conducting the usability test**

The test participants were all contacted a few days prior to the test, to check their availability and make sure that they knew where they were going. Another reason for this contact was to make sure that the participants felt welcomed and that their partaking in the study was appreciated. The actual usability test procedure was alike for all test participants. It started when the test leader welcomed the user to the test location, and by conducting normal conversation trying to defuse the tension around the actual test. Participants were then offered refreshments and were given a tour of the test area to give them a good feeling about the location where the test was given. Possible remote observers were presented to the test participant in order to give him an overall picture of the total amount of persons watching him during the test, as recommended by Barnum (2002). Then the test leader introduced the system SpatialAce in general and SpaceLab in particular to the participant, giving him a system overview picture and a preview to the interface appearance. Discussions about think-aloud protocols were also conducted, giving the participant information about the technique and a chance for him to practice before the real test began. Information about videotaping and logging software were given at the same time, which turned the attention to the consent form (appendix 4) where the participants signed their agreement that the test could be logged.

Before the test was started the test leader emphasised that it was the software that was tested and not the participant. Also given at this point was information about the participants' right to terminate the test if wanting to.

The test as well as the logging software started when the participant opened the application SpaceLab. As recommended, test tasks were read out loud by the test leader as well as given to the participant on a piece of paper for him to watch if something became unclear. The tasks were given in order of their number, concluding the test with scenario seven (all test tasks can be found in appendix 3). Following the test the participant was asked to fill in a post-test questionnaire (appendix 5) that checked how he had perceived SpaceLab during the test, but also controlling his programming skills to be able to separate programmers from non-programmers. This was done to check if the participant screening process had been conducted correctly, aiming at five participants with programming experience and five without. When the questionnaire was finished the test leader expressed his gratitude to the participant, giving him a small gift to show how appreciated his efforts was. The participant was then followed to the door (entire test manus can be found in appendix 3).

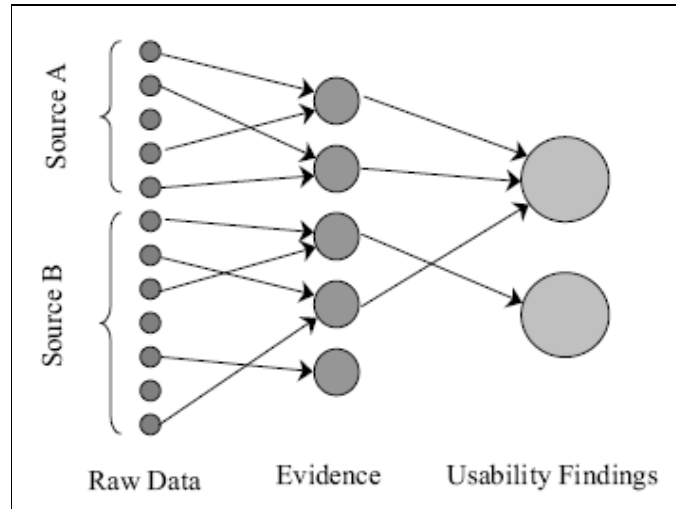
During the test some of the participants experienced problems, and got stuck at some task and at this point the test leader lead them on to the correct path. The participants were urged to try themselves at first, but if no progress was made the test leader gave them a hint what might help them, in order to get the test process going again. Questions asked by the users were only answered if it did not affect the ongoing test.

### 3.3.5 Structuring and analysing the results

The analysis phase consisted of two different steps, analysing raw data and analysing evidence of usability problems. The raw data collected in the usability tests was analysed according to the measurements in appendix 3 and evidences of usability problems were identified. These evidences were noted in each video sequence recorded during the tests, along with notions about user comments and other interesting happenings. In order to call these findings usability problems and not just a slight problem in the user interface, they had to fit into the definition of such; in this test as suggested by Ebling and John (2000) a usability problem is identified if:

- *“two or more sources of evidence suggest a problem,*
- *two or more users suggest a problem,*
- *one or more users crash the interface,*
- *one or more users identify a bug in the interface, or*
- *one user’s evidence suggests a problem and the author concur.”*

(Ebling and John, 2000)



**Figure 6.** A description of the analysis process where some raw data is transformed into evidence and some evidence is converged into usability findings (Ebling and John, 2000, p. 291).

Findings that by definition could be called a usability problem were then sorted into different categories based on a set of heuristics. The set of heuristics that were used was suggested by Barnum (2002) who claims that the Yardstick™ heuristics are perfect for usability testing. The results found in the usability tests are presented in section 3.4.3, sorted by each heuristic they are said to break.

<p><b>Clarify the core concepts</b> Does the design match the user's mental model (e.g., the desktop metaphor)?</p> <p><b>Fit content to customers who use the product</b> Does the content fit the user's expectations? Is the information organised or mapped properly and at the right level for the audience? Are there any missing parts?</p> <p><b>Plan and maintain consistency</b> Is there consistency from program to program (external consistency) and within a program (internal consistency)?</p> <p><b>Provide reassuring feedback</b> How does the system provide feedback or information to the user (dialog boxes, time to complete tasks etc.)?</p> <p><b>Clarify interaction rules</b> Does the system follow a standard set of rules for interacting with the product from a design point of view (e.g., drag and drop or buttons)?</p> <p><b>Structure navigation clearly</b> Does the user understand how to get from point A to point B (menu, icon, keyboard use)?</p> <p><b>Use plain terminology</b> Does the language match the user's own vocabulary of use?</p> <p><b>Optimise user assistance</b> Can users get help when they need it (online help or performance support)?</p> <p><b>Optimise visual design</b> Is the visual design effective and pleasing (e.g., do radio buttons look like radio buttons; are icons intuitive in design)?</p> <p><b>Design for the context of use</b> Can the product be used as designed (are there "bugs" in the system that prevent it or is the context of use poorly conceived)?</p>
---

**Figure 7.** The Yardstick™ heuristics for measuring usability findings (free from Barnum, 2002, p. 169).

### **3.4 Results of the empirical study**

This section presents the results of the empirical study, divided into the mentioned three phases.

#### **3.4.1 Interview**

During the interview with a SpatialAce support worker a lot of important issues came up, regarding SpatialAce as a whole and SpaceLab in particular. Carmenta have had their support database since 2004 and are now logging all support cases that are sent in to their support mail address. It is usually the same group of users that are in contact with the support team, probably because they know that they most often get a quick answer.

One of the most common problem with SpaceLab, as experienced by the support group, is trouble to install the environment. Many users forget that Java is required for SpaceLab to work correctly, which causes some trouble for certain users. Another case that the support workers often must help out with is how to alter a configuration file to achieve a certain goal, for instance to change colour in a polygon. Users could solve such problems by consulting the documentation, but many send their issues to the support team instead.

It is quite rare that users express themselves about their opinions regarding the usability of the product, but it has happened. Users have for instance mentioned as a usability problem the impossibility to decrease the size of the area in SpaceLab that not presents the configuration tree. In earlier versions of SpaceLab, user comments on scrolling and undo/redo buttons have caused alterations applied in the current design. A few users have urged for more use of dialogs, which they felt would ease the interaction with the system for users with less programming skills.

Other improvement suggestions made by the support worker were to minimize display and importance of programming syntax such as brackets or backslashes. These could be added in a syntactically correct way by the program itself, easing the users input. It would also be good if the documentation could include suggestions of how to organize big configuration files, and examples of such large configurations. This could be included in a new tips and tricks section for users that are reaching an intermediate or advance level of understanding for the program.

Another interesting point that came up during the interview was the fact that Carmenta does not collect any feedback from users that have bought the SpatialAce product except for what is mailed to the support team. This could mean, according to the support worker, that Carmenta does not entirely know what parts of SpatialAce that actually are used at a daily basis, or what the customers think of the product.

### **3.4.2 Heuristic evaluation**

A number of usability flaws within SpaceLab were detected during the heuristic evaluations (some even included a connection to SpaceExplorer). These weaknesses will be listed below, sorted by what heuristic they are said to break but with no internal ranking within the heuristics. Since all findings were given a major or minor problem label, these labels will also be mentioned in relation to each flaw. As recommended by Nielsen (1990) the major label was given to findings that had serious potential of confusing the user or causing him to use the system in a wrongly way. The severity of each usability problem was decided by the evaluators together and is therefore more thought through than they would be if only one evaluator would have decided on them. This was discussed together by all evaluators to enhance the possibility of reaching the same conclusion about the problems as a user would have done. The usability problems that are listed are a summary of what the evaluators found, to avoid publishing duplicate entries. In order to see an explanation to what each heuristic means, see Figure 7 above.

#### **Visibility of system status**

It is not evident what happens when text mode is chosen; it depends on the marker's position at the time of the mouse click. Minor problem.

Three evaluators mentioned that too long time passed when a configuration file was sent to SpaceExplorer before anything appeared on screen. Considered a major problem since it might confuse the user, not giving him any feedback of what is happening.

One evaluator mentioned that it was not evident what happened to the map when something was changed in the configuration file. A direct update in SpaceExplorer would be a good solution. Minor problem.

#### **Match between system and the real world**

All evaluators noted that the system is built on visualisation and programming terminology and for a user to fully grasp all information he must be familiar with such terminology. Two of the evaluators claimed the abbreviations as especially hard to understand. Major problem since the terminology are important to understand what each part of the software does, and if the user cannot understand names than he might try the wrong operator.

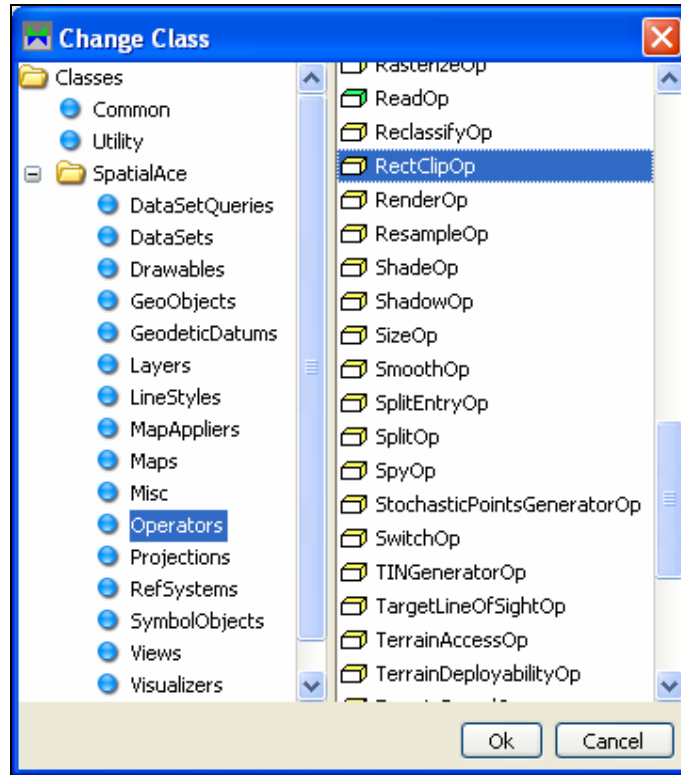


Figure 8. Example of complicated operator names

It is unclear what the check box called Freeze means, located under the tab Files. Freeze can be understood as something that locks the system, according to one evaluator. Minor problem.

The same evaluator said that objectShelf in the tools area also was unclear. Despite mouseover help it cannot be fully understood without opening the documentation. Minor problem.

### User control and freedom

One evaluator commented on the positive aspects of supporting undo/redo actions, but was uncertain of the difference between them and the forward/back functions. Minor problem.

Another evaluator was concerned that he did not know how much of his work that would be undone if he pressed the undo button. He wanted some form of list of actions of what he could undo, preferably when holding the mouse over the undo button. Considered a major problem since the undoing of too many actions will confuse the user.

### **Consistency and standards**

Under the Files tab and with a certain file is selected it is impossible to use the up and down arrows on the keyboard to navigate through the list. Minor problem.

All of the evaluators noted that an incomplete node is given the colour green, which usually is an OK-colour. One of the evaluators also said he was missing an explanation legend to all colours. Major problem since it definitively confuses the user, who probably is used to green as a colour for anything that is alright.

Another colour related issue discovered by one evaluator was the fact that different nodes have different colours, green for instance represents a ReadOp. It can be confusing for the user when same colour means different things. Minor problem.

Two evaluators reacted to the different standards in SpaceLab, some choices have dropdown menus, others have dialogs, some must be defined with quotation marks others do not. They felt that the system needs to be more consistent. Given a major problem label, since the inconsistency causes the users to use the system in an erroneously way.

When defining something that should be written in capital letters, one evaluator found out that it is impossible to erase letters when holding the Shift key. Minor problem.

The use of question mark icon for opening help in SpaceLab is evident, but in SpaceExplorer it is used to open information about SpaceExplorer. Even as SpaceLab and SpaceExplorer are different programs, it is important to strive for consistency within the program family. Using the same symbol for different actions are confusing, therefore considered a major problem

When the user is about to add a condition, for instance in an AttFilterOp, he needs to add a condition first and then add a condition again, inside the condition in order to set its properties. It is unclear why conditions cannot be set at the first dialog. Minor problem.

One evaluator claimed that it is a problem that users cannot select parts of layers (or if it is possible it is done in a way that is not very intuitive), such as two operators and move these together. Minor problem.

When working with several configurations at the same time and wanting to switch between them, the recommended shortcut Ctrl+Tab changes windows in a cycle, not between last used windows. This can, according to one evaluator, annoy users that are used to work within Windows graphical user interfaces. Minor problem.

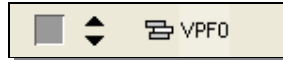
Not all lists are sorted alphabetically, for instance the dropdown menu containing different databasePahts existing in dataset: VPF needs to be sorted. Minor problem.

### **Error prevention**

One evaluator noted that the way that the different nodes are presented makes the overview a bit unclear; they should be placed in straight lines instead. Minor problem.

It is possible to enter a value in a condition that cannot be used, for instance a negative when the condition only handles positive values. Minor problem.

According to one evaluator there is a problem when closing SpaceLab while having objects placed on the object shelf. If the marker is not placed on the object shelf no question if the user wants to save the shelf objects will turn up, and they will therefore be lost at the next launch of SpaceLab. Considered a major problem since the lack of feedback can cause the user to loose data.



**Figure 9.** The object shelf

Two evaluators claimed that when a user wants to see the configuration file he is working on he must save the file since SpaceExplorer only views the latest saved configuration. No information is given about that, SpaceLab only asks if the user wants to save the file or not. Since the user do not get the information that he has to save the configuration file to see any changes, this problem causes confusion and are therefore labelled a major problem.

The icons that are used to change node sizes do not inform the user of what nodes that is affected of the change. It should somehow show that it is dependent on where the user marker is. Minor problem.



**Figure 10.** Icons for node size changing

### **Recognition rather than recall**

Two evaluators reacted on the listings in the folders Common and Utility, what those include is not evident to a new user. Users should not have to guess what a folder contains. Minor problem.

Another of the evaluators claimed that it was hard to understand what the different objects in the folders were and that users had to remember their functions by themselves. Since the users have to look for each object in the whole class library, the unclear sorting of objects are clearly a major problem.

### **Flexibility and efficiency of use**

The folders Common and Utility include a certain classes and objects. These objects should be chosen by the user, since what is a common object for one user might not be it for another. And as the system supports drag and drop actions that would be an easy way to change the contents of the folders. Minor problem.

### **Aesthetic and minimalist design**

When adding a condition to an operator it can be a bit unclear what the user is supposed to do, the design of the dialog is too minimalistic and confuses the user. Minor problem.

### **Help users recognize, diagnose, and recover from errors**

All of the evaluators reacted strongly against the error messages that SpaceLab generated. They included lots of programming language and no evaluator actually understood what they did wrong based on the information in the error message. No information was given about how to avoid the erroneous behaviour. Using code in error messages is very confusing for users and this problem were therefore given a major label.

### **Help and documentation**

Two of the evaluators noted that the first example configuration (called Hello World) in the documentation had flaws. The pictures are referred to at wrong locations and information about why certain properties are chosen have been left out. Especially mentioned was why `featureClass` should be set to `POAREA`, no information about that is given. Another thing that upset one evaluator was the use of quotation marks when defining for instance `featureClass`, which was not described well within the documentation. Minor problem.

It was hard to get task based help according to one evaluator, for instance information about what a certain operator could do, without having to select it. The evaluator said he was missing mouseover help labels on objects within the software. Minor problem.

## **3.4.3 Usability Test**

The results of the usability tests present usability problems found and analysed as described in section 3.3.5. The findings is founded on the ten planned usability tests but also, when applicable the dry run test and the pilot test, since these sessions also contributed with important input to the evaluation process. In those cases where the test leader considered a usability finding possible to sort into two different categories of heuristics, the one found best fitting were chosen.

### **Clarify the core concepts**

Four of the test participants had problems with understanding the basics in the application, that a node are supposed to be chosen in the class library and then dragged and dropped in the dataflow window. The same kind of problem appeared when a few users had trouble in understanding that they needed to drop an object on a node in order to add that object to the node. Not until that action has been done is it possible to change the attributes for that object. Some participants tried to alter the attributes by only select an object in the class library and trying to change the values without dropping the object somewhere in the dataflow window.

A couple of users were confused with the appearance of the software, confusing the class library with the list of different objects currently in use presented at the Files tab. This confusion led to big frustration when they could not drag and drop objects to the dataflow window as they had been told in the introduction. Another user simply did not understand what purpose that list had, and thought it was unnecessary.

One user did not understand that SpaceExplorer only was used for previewing the configuration file; he tried to alter attributes from within SpaceExplorer itself.

### **Fit content to customers who use the product**

Almost all of the test participants had trouble with understanding the buttons that initiate different dialogs, for instance the custom dialog (...) and text editor (T) buttons. But when asked which button they would choose first from the above mentioned, nearly all participants said that they would press the (T) button because that would probably give them a chance to write something. One of the users said that he thought the most important button (in this case he mentioned the custom dialog button) should be placed to the right, because that was what he was used to.

Another issue that invoked alike responses between the participants was when they were asked to minimise the nodes in the dataflow window. Almost all of them tried to find some function within the dataflow window to complete the task, ending with them pressing each node individually. Using the right click menu was also a popular attempt to solve the problem combined with efforts trying to select all nodes at once and then choosing the appropriate menu alternative. Only two out of twelve participants found the minimise function in the top menu directly.

Most users tried to right click on different objects in SpaceLab and they expressed a wish for more functionality within the right click menu.

### **Plan and maintain consistency**

One participant experienced problems when using, a command in windows that he found normal (Ctrl+A), that selects all objects in a given area. This puzzled him quite a bit, but not as much as the failed attempt to delete a node from the dataflow window using the keyboard button delete. Having a number of unwanted nodes he successfully deleted all but one with the keyboard, but the last one he had to delete by pressing right mouse button and choose the delete alternative at the menu. He had absolutely no idea why he was forced to change his way of interacting with the software.

Other functions that normally works within software of different kinds is when a user have selected an object in a list and then presses a key on the keyboard at which point the selection jumps to the section of objects starting with that letter. This did not work for a test participant, making him both annoyed and fatigued from all scrolling he had to do.

When looking for a certain section within the help file, a great number of the participants used the index with a disappointing result. The string they searched for was not listed in the index, and no information was given why.

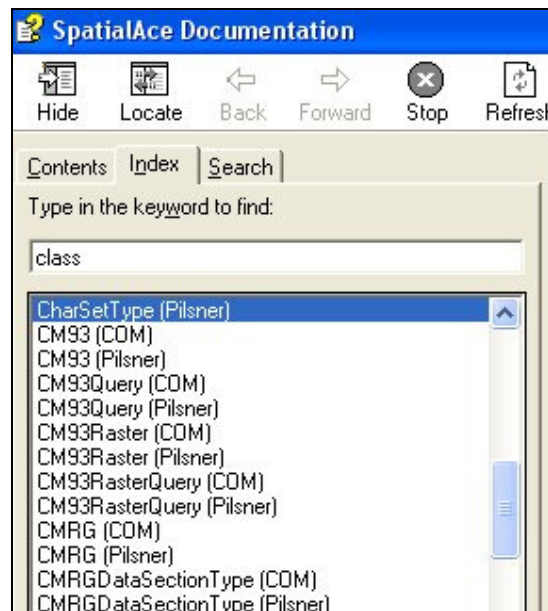


Figure 11. Index tab in help section

The part of the test that brought most agony to the participants was without a doubt the inclusion or exclusion of quotation marks. One way or the other all test participants experienced problems or uncertainty relating to the usage of quotation marks. Depending on how an attribute was altered, from where it was altered and the core type of the attribute changed, different quotation mark alternatives turned out to be correct. By having a number of possible ways of changing the attribute, but with no consistency in the insert format, the users got more and more confused while trying to solve the problem.

Four of the users found it strange that a drop down menu was presented in an attribute row despite the lack of correct attribute in the drop down list. They meant that such a list confuses the user if the right value is not available for selection.

A few participants reacted to the placement of buttons, for instance the drop down button can be presented on the same place as the custom editor button when that is not used. That made the users mix up the functionality of the two buttons. One participant thought it was strange that the buttons were not visible at all times, because he felt uncertain where there might be buttons and where he had to rely on himself to know the correct action.

A problem that caused almost as much confusion as the quotation marks was the inconsistency in the software when users were trying to choose colours in different nodes. In one node the attribute row had a preview of the chosen colour, others lacked that preview. One node required only one button to be pressed to change colour, others demanded the user to open two different dialog windows to see the same colour options. Another colour problem struck a participant when he by accident chose transparent in the colour editor, which then automatically removed the drop down

menu button making a return to RGB values impossible from that menu. The only way of solving the problem, which he could think of, was the undo function Ctrl+Z.

A couple of users tried to save a configuration file with a file name containing brackets or spaces, which the software would not allow. This was properly explained by an error message, but if other programs can allow spaces it should be possible to handle them in SpaceLab as well.

Five users commented on unsorted lists, for instance when looking for a databasepath or a reference system, they reacted to a long list of examples that were not sorted alphabetically, which prolonged their search time extensively.

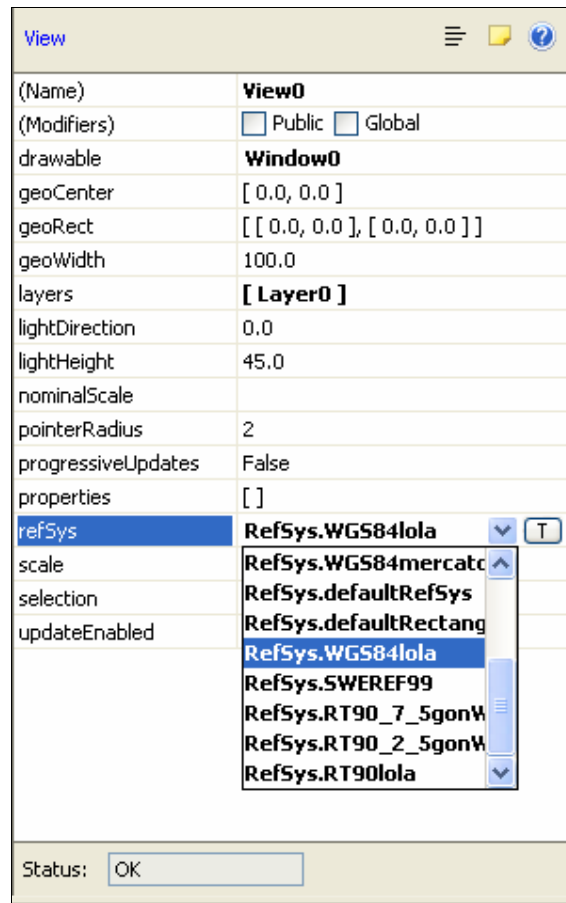


Figure 12. List of unsorted reference systems

### Provide reassuring feedback

One of the users experienced problems when he chose to right click on an object in the dataflow window and tried to change class on the selected object. He chose a new class but nothing happened, and he received no feedback about what mistake he had done or how he could avoid it in the future.

Eleven of the participants thought the error messages in SpaceLab were hard to understand and in some cases they gave no useful information at all. In relation to

this, a couple of users asked for some form of compilation function that could point out what mistakes they had done and where these were located.

Almost all participants had difficulties in seeing or understanding the feedback and coding of an incomplete/conflicted node. The users thought that a green node should represent a node without faults, but all agreed that a flawed node should have a different colour than the other in the dataflow window. One colour that was discussed was red, which would create a logical connection to the small red boxes with white crosses marking an attribute row containing a fault. These crosses were useful to the users when discovered and understood, but some participants had trouble noticing them and understanding their function. One participant also claimed he did not understand if they were directly updated when he made a change in the attribute row or if he had to save changes first, as he had been forced to when previewing a map in SpaceExplorer.

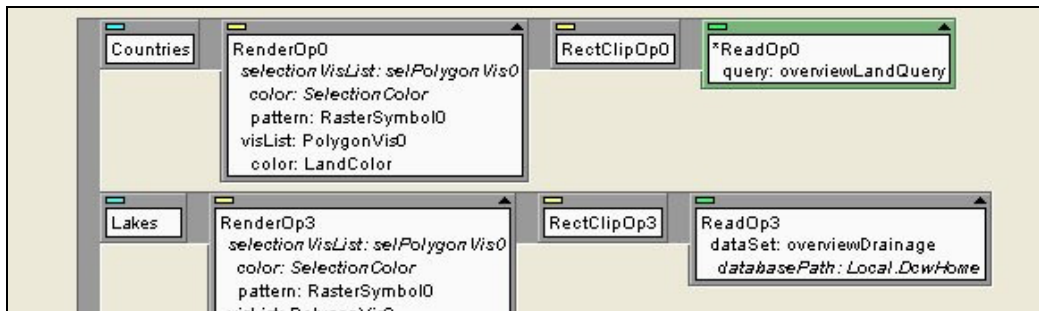


Figure 13. Incomplete node coded in green



Figure 14. Undefined dataset

One user tried to use the “edit as text” function when handling a value in the AttributeVariable editor, but his changes were not applied when pressing the ok button. Without any feedback his choice were neglected and he had no idea how to improve his interaction, if even possible.

Ten users had trouble with understanding the dialogue related to visualisation depending on key numbers. Most of them thought that the dialogue was clearer than just the text editor, but it would still need more explanation to enhance user understanding. They did for instance not understand what the difference between Key and Value was, and asked for tool tips and a clearer distinction between the different alternatives.

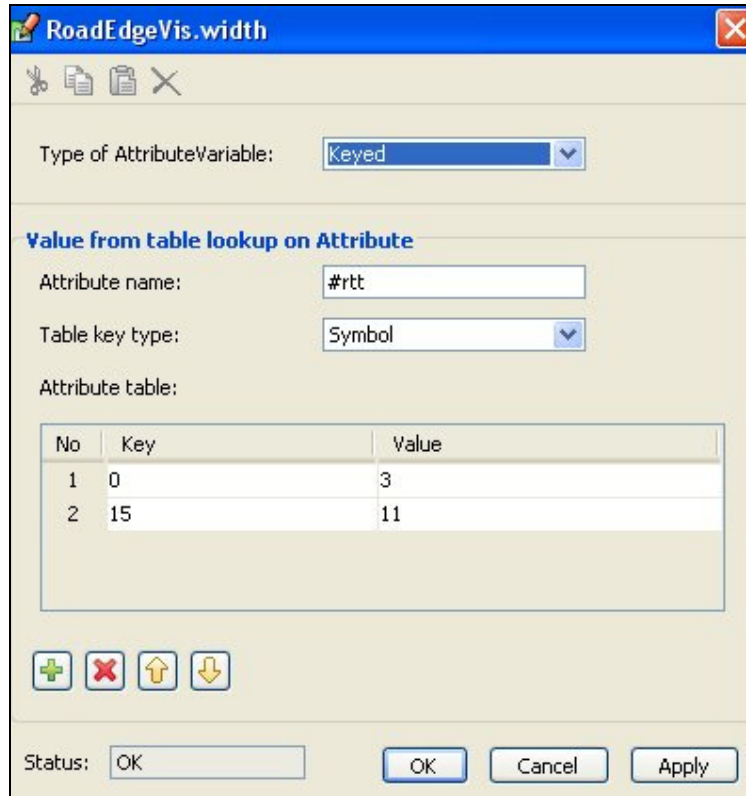


Figure 15. Editor for attribute depended visualisation

One user who discovered the toolbar alternative for changing sizes on the nodes did not understand that the function worked differently depending on the object selected. If a node were selected, the action was taken for only that node but if anything else but a node was selected the action was implemented for the entire tree of nodes. He did not understand this and found no feedback that informed him about that fact.

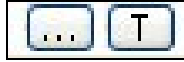
Two users reacted on the fact that they did not get any feedback about needing to save the configuration file before being able to view it in SpaceExplorer. A question if they wanted to save was presented but it contained no information about the need to do so to be able to see the changes made.

### Clarify interaction rules

A few of the participants had trouble with understanding the search function in the help section. When they searched for a string, for example classes, they expected a result including the word classes instead of a list of the headlines of the section containing information about classes. Furthermore they did not understand that they needed to select the appropriate headline in the list and double click or clicking display to make the help section appear in the right hand window. Another feature they did not notice or understand was the ranking system, since they wanted the search result to include the search word they could not understand how something without the search word was ranked at first place.

Eight users expressed a wish for or looked for a confirmation button when they had changed a colour within a node. They were uncertain if the colour they had pressed really got selected. Some of them tried to press the node status box (which could be mistaken for a button) indicating OK, and thought that would confirm their selection.

Seven test participants did not understand that pressing the (...) and (T) buttons would generate a dialog box where they could make selections. Since they did not recognise the appearance of the buttons they could not understand their meaning, forcing them to change the attributes in other ways.



**Figure 16.** Buttons to open editors

One participant misinterpreted the arrow buttons in the key table dialogue; he thought they were used to change the values in the table instead of moving the rows up and down.

Four users got frustrated when they were creating a new configuration file and added a view node but found that it was placed at a position they did not like. And when they tried to change its position it could not be moved. Two users did not understand directly how they should connect the nodes to each other; they did not see or understand the visual feedback given to enhance the connection process.

One user experienced problems when he tried to double click a query in a node, and expected some form of response from the software but found out that exactly that action in that place did not have any affect on the program.

One participant noted the strange behaviour of not being able to erase text that had been entered to an attribute row when holding the shift key.

### **Structure navigation clearly**

Within the help section one user had trouble seeing the tabs: Contents, Index and Search. Therefore he also mixed up the expected result after searching for classes, instead of a list of headlines he thought he would be presented an indexed list with classes selected.

Six users had problems when they were supposed to change colour in a node containing both a selection colour and a normal colour. Since they all investigated the node starting from top to bottom they found the object called Color:SelectionColor and without giving it any more thought they changed that attribute, not knowing that the wanted colour attribute was further down the list.



Figure 17. RenderOp with two colour objects

About one half of the participant crew found it hard no navigate through the class library and finding where different nodes and objects are located. A few users were confused by the choice of icons (round blue dots) for the folders under Classes, SpatialAce and Plugins in the hierarchy; they did not understand directly if those icons contained any additional objects.

One user got really confused when he was trying to change a value in a key table, because when he clicked in the column named value the pane in the column key at the same row was selected.

**Use plain terminology**

All users either commented on or had trouble understanding the programming related terminology in SpaceLab. They felt that it was hard to understand the purpose of all nodes especially those who were named with abbreviations.

Almost all users got confused when they were about to add a Window to the View, with problems ranging from not finding the Window object to not understanding what a Window were. Some users mixed up the object Window with the dataflow window and did not understand that they had to find an object called Window and drag it to the correct node. In relation to that problem some users did notice that something was missing in the attribute drawable, but they did not make the connection to the folder drawable in the class library where the object Window could be found. In stead they looked for objects called drawable in the class library, and with no such object to be found they got even more confused.

A couple of users had trouble with understanding the meaning of the word Nil, that were placed at some attributes. There was uncertainty of why some attributes were equipped with a zero value string when others were left totally empty?

Nine of the users expressed their confusion about what was shown in the attribute row for something that was visualised depending on its key values.

style	
width	[ key: #rtt table: [ [ 0, 3 ], [ 15, 11 ]...
worldCoords	False

Figure 18. Key table code

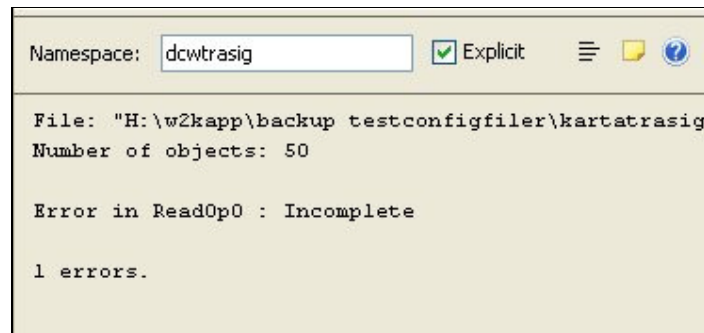
One user who accidentally changed the key numbers in a key table got the error message that the integer key values in an intkey table was not given in order, which

was kind of hard for him to understand since he did not realise what values he had changed.

### **Optimise user assistance**

One user tried to right click on the folder classes and expected an option that could help him explain what a class was.

When trying to discover what was wrong with a configuration file used in the test, one user tried to click on the error message at the lower left hand corner wishing for a response where the software would highlight the erroneous part in some way.



**Figure 19.** Clarification of the location of an error

In relation to the problem with database paths not being alphabetically sorted, one user tried to type the first letters in the path believing that such an action would exclude the options that did not match the written letters.

One user expressed a wish for more tool tips when he hovered the mouse over an object, this would according to him increase the possibility to perform the correct action.

A couple of users got confused by the way the help text was formulated in the example Hello World. One of them thought it was strange that the first sentence (that explains what are about to be done) not was differentiated from the step by step instruction how to perform the action. He thought that the headline should be made italic so it could easily be separated from the rest of the instructions.

### **Optimise visual design**

One of the users wished for usage of tabs when working with several configuration files at once. That would ease the process of switching windows, letting the user just click a tab instead of choosing a different file in the windows menu

### **Design for the context of use**

All users experienced problems with sending a configuration file to SpaceExplorer, the function did not work all times and there were not always feedback informing about the system status.

### 3.4.4 Post test questionnaire

The post test questionnaire that was given to the test participants directly after the test was finished provided some interesting results. Noticeable is the fact that the participants finally performing the test did not quite match the characteristics chosen in section 3.3.2. Only four of the original ten participants (five was the number sought for) turned out to possess the required amount of programming skills. This can be explained by the participants' different opinions of what constitutes good programming skills, and also a slightly unclear question about their programming experience in the first phase of the participant selection phase. But when adding the opinions from the dry run test and the pilot test, the sought number of programmers was achieved. The number of non programmers reached seven, but that was only considered positive for the study since all opinions of the software were beneficial for the thesis. The fact that the number of programmers and non-programmers were uneven did not matter that much, since no statistical generalisations were to be made. The opinions about the application and the participants' improvement suggestions for SpaceLab are listed below. Table 1 show the impression of SpaceLab as perceived by the test users after interacting with the system for about an hour. Each column contains the number of participants that choose that alternative out of a total of twelve users, including the answers from the participants of the dry run test and the pilot test. (The original chart was in Swedish and can be found in appendix 5.)

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	Don't know
SpaceLab is easy to use	1	2	2	6	1	0
The feedback is sufficient in SpaceLab	1	2	5	4	0	0
SpaceLab is hard to learn	4	3	4	0	1	0
It is easy to get lost in SpaceLab	2	5	0	3	2	0
It takes training to understand SpaceLab	0	0	0	9	3	0
The help section is usable	0	0	5	5	2	0

**Table 1.** Summary of user's impression of SpaceLab

Below are two parts of same chart divided into programmers' and nonprogrammers' opinions, showing dry run participant and pilot participant are marked with + 1. Note that in the non-programmer table there are seven participants compared to five in the programmer chart.

## THE EMPIRICAL STUDY

---

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
SpaceLab is easy to use				3+1	1
The feedback is sufficient in SpaceLab		1	1+1	2	
SpaceLab is hard to learn	3	+1	1		
It is easy to get lost in SpaceLab	2	2+1			
It takes training to understand SpaceLab				3+1	1
The help section is usable			1+1	1	2

**Table 2.** Summary of programmer's impression of SpaceLab

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
SpaceLab is easy to use	1	2	1+1	2	
The feedback is sufficient in SpaceLab	1	1	2+1	2	
SpaceLab is hard to learn	1	2	2+1		1
It is easy to get lost in SpaceLab		2		3	1+1
It takes training to understand SpaceLab				5	1+1
The help section is usable			3	3+1	

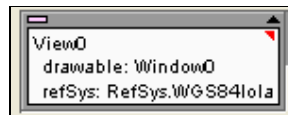
**Table 3.** Summary of non-programmer's impression of SpaceLab

In the post test questionnaire the users were asked to list three different improvement suggestions to facilitate the learning process for new users and one of the most prominent proposition involved changing the colour coding for incomplete and erroneous nodes. Other suggestions that were mentioned by several participants were: change the vocabulary within the software to avoid programming terminology, a clearer first example in the help section where Hello World is now, and clearer feedback about what attribute that is incorrect when an erroneous node is selected.

Following suggestions were mentioned only once in all the questionnaires: An OK or apply button when an attribute have been changed, remove the need for users to keep track of which attributes that requires quotation marks – it should be handled by the software itself with no concern of the user, develop the feedback function that is shown in the lower left corner of the software – providing feedback that is easier to understand, clearer categorisation in the class library facilitating user understanding of

what each category includes, a tip of the day when the software starts including some recommendation that makes the daily work with the software easier, usage of tabs when working with several configuration files at once, usage of basic configuration files including the most common nodes – so the user can start his map configuration without having to add these nodes every times, and a better introduction to the system including a question if the user would like to go through a tutorial the first time the system starts.

Other suggestions that were given were to improve the error messages – giving information about what was done wrong and how it can be avoided, more functionality within the right click menu, more distinguished toolbar buttons for changing size of the nodes, a light version of the help section for rookie users, displaying results containing the search phrase when searching for a section within the help – instead of showing the headline for a section containing the search phrase, change the symbol/symbol colour for a commented node, change the appearance of the button that launches the custom editor, a legend that explains what all node colours mean, better explanations/help within different editors, show in a clearer way how SpaceExplorer is started, include a search function in the class library so users can search for a node or object they can not find, and give the user a possibility to chose the location of different parts of the software – for instance to move the class library to the right hand side of the screen.



**Figure 20.** A node with a comment

### 3.4.5 Result summary

This section tries very briefly to sum up the results of the empirical study, listing the major usability problems found in all parts of the evaluation.

The study show that SpaceLab contains quite a few areas where improvements are possible, but the most important ones seem to be: The usage of programming terminology, the very brief introduction to the program, the lack of feedback when sending a configuration to SpaceExplorer, the inconsistent usage of quotation marks in attributes, the colour coding for erroneous or incomplete nodes, overall feedback indicating where errors are, the lack of consistency within the software when conducting related choices, the unclear and no help providing error messages, the lack of functionality within the right click menu, the usage of the index tab within the help section, overall dialogue button appearances, the way a confirmation of a made selection is made and the unclear Hello World example in the help section.

## CHAPTER 4

### DISCUSSION, RECOMMENDATIONS AND CONCLUSION

---

*This chapter connects the different parts of the study and gives possible answers to the research questions found in the first chapter. The results acquired in the third chapter is interpreted and related to the theoretical research framework. Also included is a discussion about the research process with notions on what have been successful and what could have been done differently. Following the discussions are the recommendations for design alterations. The chapter ends with conclusions and recommendations for future research.*

---

#### **4.1 Discussion**

The discussion is divided into two different sections, starting with an overall discussion about the research process and the second part serves to answer the first parts of the research questions and to reflect on the results of the empirical study. How the found usability problems can be eliminated will be presented in 4.2 recommendations.

The study was conducted to answer the research questions:

- What problems do new users to SpaceLab experience and how can those problems be reduced?
- What are the overall usability problems within SpaceLab and how can they be reduced?

However, also given some extra thought was:

- Is there a difference in how new users with programming experience understand SpaceLab compared to new users without programming experience? What is that difference?

### 4.1.1 Research process

In order to gain a deeper understanding of how to be able to answer the research question the research process started with a literature study including human cognitive limitations and usability testing combined with an introduction to SpaceLab. Thereafter it was time to decide the methods for the empirical study. Considering the aim of the thesis and the fact that Carmenta provided the test logging software Morae, it was easy to choose a classic usability test for the main part of data collection. This choice agreed with the opinion of Wichansky (2000) who said that there is no substitute for actual user data when trying to decide if a program is usable. The selection of usability tests meant a qualitative approach to research that, according to Holme and Solvang (1996/1997), would give an overall picture of the problem at hand. But in order to get background information about known problems within SpaceLab an interview with a support worker seemed like a good base, especially since she could get hold of all SpaceLab support cases reported the last year. By reviewing literature I decided to use a semi structured interview with note taking as documenting method. The notes were worked up directly after the interview session as recommended by Kvale (1996/1997), and since the interviewee worked at the same office I felt no risk at losing any material as it would have been possible to recap what said at any time.

Following the interview four heuristical evaluations were conducted and as Nielsen (1992) claimed, both major and minor usability problems were found. One problem that occurred during this process was the labelling of some usability flaws, with the different evaluators having separated opinions. Some compromises had to be done, which might have ended with an erroneous label given to some usability problems. As described by Barnum (2002) the result of the heuristical evaluation can be of great use when constructing the tasks for the actual usability test, and that was also the case in this process. The expert evaluations gave a good overview of what parts of the software that would benefit most from being tested, and therefore some tasks for the test were developed including interaction in those areas. The heuristical evaluation might have provided another result if the experts had been given different tasks, but it is my conviction that such a result only would have differed slightly since the main areas of SpaceLab were covered in the original tasks. In addition to the given assignments, the evaluators were allowed to interact freely with the interface during a sufficient amount of time, as suggested by Nielsen (1992), which allowed them time to discover other usability issues.

The usability tests process followed recommendations as mentioned in section 2.7, starting with the writing of a test plan. When writing the test plan it was a goal to make sure that the test would be sorted into the category that Dumas and Redish (1999) would consider a real usability test, containing five mentioned characteristics; the goal must be to improve the usability of a product, the test participants represent real users, the participants perform real tasks during the test, that the test leaders observe and collect data during the tests, and that the test leaders analyze the collected data and recommend improvements to the product. I think that these characteristics are well complied with, and the result therefore can be considered when deciding if the software are usable.

One small problem in the test process was deciding on how many test participants that would be enough to claim a sufficient amount of data, but following Dumas and Redish's (1999) recommendations I decided on two different groups of five users each. The most important quality in the considered users was the fact that they were correct and representative to the target audience, with no earlier experience from the software. What could have been done differently at this phase was a more thorough investigation of the participants programming experience, but with them all having different frame of references of what constituted good programming skills, that might not have made any difference anyway. It ultimately turned out satisfactorily anyhow, since the dry run and pilot test results could be added to the overall result.

The construction of the test involved no major difficulties, but a great focal point in this work was the try to cover both typical and critical tasks. Nielsen's (1993) suggestions about starting and ending the test with rather easy tasks were also followed. Another aspect that limited the test length was the approximate time for completion of each task, aiming at all participants finishing the test tasks within an hour. These estimated times for each tasks were built on the authors own experiences with the interface, and thoroughly checked during the dry runt and pilot tests. Test participants for the dry run and pilot tests were recruited to suit the target audience, if the tests would go without major problems, which they later turned out to do. Small corrections to the test were made after these sessions, but nothing that affected the gathered material that instead was added to the overall result. Both these tests were very useful to the test leader as indicated by Barnum (2002), giving him experience and training in speaking from the script.

As Nielsen (1993) pointed out, it was needed to introduce the users to the system before the test started. The brief introduction given proved valuable to the users, and a number of them referred to the introduction when trying to solve a test task.

The test sessions with the original ten participants went incredibly smooth, really only containing two small setbacks in all ten tests. These setbacks were both outside the test leaders control, the first included an unexpected termination of the logging software and the second contained a lost connection to the network. First problem was solved by simply restarting the software and continuing the test, and it later turned out that the material recorded before the crash could be restored, which meant that the crash did no major harm to the test data. The second problem got solved when the test leader switched order for two test tasks, abolishing the need for network connection. During the tests some of the participants had trouble remembering to think aloud, whereupon the test leader intervened and politely asked the user what he was thinking of. This behaviour was not recommended by Barnum (2002) but worked perfectly during these tests, and no participant reacted strangely to those questions. Instead they improved their think aloud usage which ultimately enhanced the test results. Another discovery made during the sessions (also pointed out by Barnum, 2002) was that it cannot be said clearly enough to the participants that it is the software being tested and not their skills. Many of them showed signs of feeling stupid or nervous when failing to complete a task, especially when they knew someone was observing from another room.

Following the test session all participants were asked to fill in a questionnaire, to gather their opinions about the software. Included in the questionnaire were a couple

of questions concerning the participants' programming experience. As described earlier this area caused some trouble during the entire test process, so also within the questionnaire. The questions asked the user to rate his programming knowledge from non-existent, small, large to programmer and gave an empty space for him to fill in his number of academic points in programming. Several users had trouble in translating their knowledge into one of these options and felt that the question was hard to answer, and I could not agree more. The question was badly formulated and I should have used other choices for them to choose from. A number of users also had problems with estimating their number of academic points in programming since it often is included in other courses, a fact that made them unsure if they could count those points. Clear rules of what could be counted should have been set, or maybe a totally different way of deciding their programming experience would have been used. Except these issues the questionnaire worked as intended, giving some useful responses and suggestions. By following Barnum's (2002) recommendation to avoid too many open answer question, the time of analysing the questionnaire could be limited.

The analysis phase of the research process was conducted directly following the test phase. Initially the material from the logging program was studied and evidences of usability problems were noted. These evidences were thereafter compared to what was found in the heuristical evaluations and the interview, and those evidences that could be considered usability problems according to the mentioned definition were then sorted into a category of Yardstick™ heuristics. This part of the analysis was quite hard since many usability problems seemed to fit into two or more categories, but that was solved by choosing the one found most suitable. Using the top-down approach to analysis suited this thesis well, and kept the researcher alert for other usability problems then just the expected, as predicted by Barnum (2002). The outcome of the analysis is presented in section 4.1.2.

Concerning the validity of the study I think that I have done what could have been done to assure a high validity. The test participants were all matched to the software target audience, the test consisted of both typical and critical tasks as suggested by Holleran (1991). The question if the study is reliable is harder, since the definition of reliability is that a collection of measured data on one occasion must match the same measurement another occasion or by another researcher. Analysing collected data is a process that demands a lot from the researcher and I think that what each researcher brings into such a situation affects how he interprets data. This is also mentioned by Eysenck and Keane (2000) who emphasize the affect of earlier experiences in any information processing situation. For instance, another researcher with more experience from SpaceLab might interpret the data collected in this test differently from my analysis. But I do not think that such a theory can be said to reduce the reliability of this study, because to me a reliable study is when something is performed from the same prerequisites which would not be the case in that theory. By giving the test participants the same starting conditions, that all test were performed according to a test script as recommended by Holleran (1991), I think that I at least have secured a great possibility for high reliability.

An overall impression of the research process is that it has all gone very well. I think it is impossible to go through a process of this size without small setbacks, but I am very content with how the work has been carried out. Another notion that comes to

mind, is that I feel that my design recommendations, presented in section 4.2, are well founded, based on literature studies, an interview, heuristical evaluations, usability tests and the users own suggestions for improvements.

### 4.1.2 Empirical study

The results obtained in the empirical study are discussed in the following section, including all three phases of the study: interview, heuristic evaluations and usability tests. Instead of sorting the usability problems based on in which part of the study they were found, this discussion will sort them depending on the author's view on how serious they are. Not all usability problems are discussed, but the ones considered most interesting are represented. The only part of the results that are separated from the others are the post test questionnaire result, since that concerns the test participants' overall opinion of the software instead of listing a number of usability problems.

The most serious usability flaw discovered during the entire research process, experienced by all test participants and mentioned by three of the experts, is the problem when sending a configuration file to SpaceExplorer. When the users pushes the SpaceExplorer button, he receives no feedback whatsoever about the system status. He does not know if he has initiated the viewer or if the system is idle and just waiting for him to perform an action. This indistinctiveness by the program can cause major irritation and frustration with the user, and goes clearly against what Norman (1988) mentions, that it is fundamental to understand what effects user actions have on the system. If it is not possible to understand what a system is doing, neither me nor Preece et al. (2002) can say that such software have easy user interaction.

Barnum (2002) claims that a product being developed should at least make sense to the potential users, or it might give them an erroneous mental model of the system. Having a faulty mental model can cause frustration and strange behaviour. Almost all test participants and experts that came in contact with the system during the process experienced frustration over the fact that they did not understand the terminology of the system. I think that all strange names, abbreviations and operators confuse the users and make them unsure in their interaction with the system. Preece et al. (2002) claims that in order to provide a clear system image for the users to communicate with, the system must be easy to understand and have intuitive interaction methods. I think that if users can not understand what certain part of an interface does, then it does not matter how intuitive the interaction is since it will not be used voluntarily anyway. This problem also includes confusion involving words with several different meanings, such as window. The fact that SpaceLab contains an often used object called Window brought a great deal of confusion and frustration to the users. Window is a term used often in all kinds of computer related contexts, which indicate that it has several meanings only in that area. Therefore help and instructions related to the SpaceLab Window object must be crystal clear to avoid confusion, something that is not the case today.

A related problem is that the users had problems with finding different objects in the class library, an issue relating to the labelling of folders and objects. SpaceLab

contains folders that for instance are called drawable, and with no programming experience it can be hard to understand that the Window object can be found there. But one good thing in the class library is that all objects in the different subfolders are shown in a list if the user selects one of the main folders Classes, SpatialAce or Plugins. Despite this resulting in the lists of objects become longer, it helps the users to find the sought objects. Preece et al. (2002) emphasize the importance of easiness to understand and intuitive interaction methods, which are, regardless of the forgiving alternative of choosing a main folder, not present in SpaceLab at the moment.

Nielsen and Molich (1990) claims that good interface design should prevent errors from occurring, and in software a good way of accomplishing this is to be consistent with design choices. A large part of the test participants and two of the evaluators had problems with consistency in the usage of quotation marks. I think that the fact that some attributes requires quotation marks and others do not causes the users to focus an unnecessary large amount of their attention to the change of an attribute, an action that should be done without any extra effort. When focusing the attention to a certain part of the software, what Eysenck and Keane (2000) call the attentional spotlight, everything outside the spotlight is really hard to see. At the time when the user wants to switch back to an overview of the software again the change causes an unnecessary time delay in the work.

Although error prevention is the best solution in design it is impossible to avoid all thinkable faults, and when those occur it is important to help the user to recover by providing good error messages. Preece et al. (2002) mentions the importance of providing the user with information about the error, how to recover from it and what to do to avoid it in the future. The participants in the study did not experience that kind of help from the error messages in SpaceLab, instead they got frustrated that they did not understand anything. A few of the users could guess in what area a problem was from reading one of the messages they received, but overall the error messages in SpaceLab caused more annoyance than help.

One thing that most users in the tests noticed was the colour coding for an incomplete or an erroneous node. Using a colour to separate an object from others is recommended by for instance Hertzler and Novell (1998) who says that it is the single most effective code when considering visual reaction times. The problem within SpaceLab is that the colours used are not the ones expected, coding an incomplete node with the colour green and a node with a conflict in a pink colour. Meier (1988) claims that a colour coded system that uses the coding in a faulty manner actually is worse for the user than a system without codes. I think that using green to signal that something is wrong confuses the user, since green usually is a notion for something to be okay. This doubles the user's workload by first forcing him to understand that green does not represent what he thinks it does, and thereafter making him search for what might be wrong within the node. The attempt to give an erroneous node a distinguished characteristic is well thought through, but as Nielsen and Molich (1990) claims, it is also really important to follow conventions and standards. A feature that does have the right colour is the red box with a white cross at the start of an attribute row when there is something wrong with that attribute. But many of the participants did not notice that cross, since it is rather small and in the periphery of the users' visual attention span. The symbol is barely noticeable and too small to mark a distinct difference compared to the other attribute lines. By doing this and not give it an

enough visible characteristic the good qualities in colour, as claimed by Tufte, (1990) are missed out on. Another small and barely noticeable important feedback is the tiny star within a node showing what object that contains something erroneous. Since it is the same colour as the rest of the text it is hard to observe and the Law of Similarity, as mentioned by Coren et al. (1999) is not broken. A quick scan of the node might help the user to see it, but at least half the participants in the test did not notice it. I think that the star issue contains the same problem as the colour choice for an erroneous node, it is good to try to separate it from the other objects but it must be done more clearly.

As stated above and mentioned by Barnum (2002), a very important issue when designing a program is to maintain consistency. Preece et al. (2002) mention the importance of clear interaction rules, and in my view these two opinions are good to follow to ease the user interaction. Unfortunately that is not always the case in SpaceLab, having different ways of selecting colours at different locations in the software. This is probably because the usage of attribute dependable colours, giving items different colours depending on key values but it is my conviction that the first dialogue box always should look alike. If users are presented with the same dialogue at all times when dealing with colours and in that box then choose if they want to use attribute dependable values, they will be more confident in their interaction since they will recognise themselves from previous actions. This might not be the best option if the attribute dependable alternative is used a lot more frequently, but in order to know that for sure, a larger study about user habits must be performed.

Except from maintaining consistency within and between programs it is recommended that accepted standards are followed, among authors that emphasize this are Nielsen and Molich (1990) and Preece et al. (2002). During the tests several breaches against such standards (that were at least accepted by the test participants) were discovered. A couple of examples of this are the non acceptance of spaces in filenames and the impossibility to use erase when holding the shift button. Such confusing behaviour by a program can cause serious frustration with the users, if an action they usually do in a program does not work within another. Since the SpaceLab environment strongly reminds of other Windows based applications I believe that most users think that it will behave like other known programs. Eysenck and Keane (2000) mention to the importance of earlier experience and the context in which an action happens, and that it clearly affects the information processing in that situation. These kinds of problems caused by not following standards seem really unnecessary to me, since just sticking to the accepted standards would abolish them altogether.

Another functionality that is more apparent in other programs compared to SpaceLab is the usage of the right click menu. Almost all users tried to use that function at some point during the tests, but had to find other ways of completing the tasks since the right click menu in SpaceLab contains very few functions.

### **Post test questionnaire**

The post test questionnaire results indicates that a majority of the test participants consider SpaceLab easy to use, quite easy to learn and that it is not that easy to get lost in the program. Relating to the question of feedback the result can be considered more neutral, including both positive and negative reviews. Most users also think that

the help section is usable, and all of them think that it takes training to understand SpaceLab.

These results are quite interesting since they do not ultimately reflect the findings in the usability tests or the heuristical evaluation. Despite a long list of usability problems found most users consider SpaceLab easy to use. This can be explained with a closer look at the figures for programmers and non-programmers separately, that clearly shows that programmers find the program more usable than non-programmers.

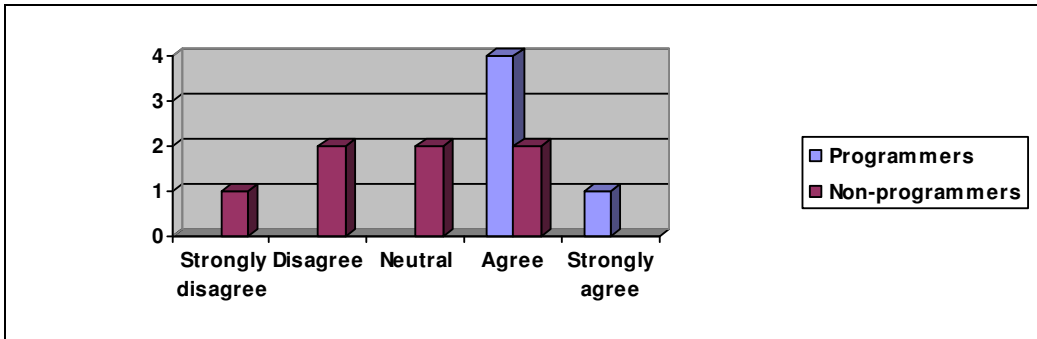


Figure 21. Level of agreement: SpaceLab is easy to use

Without jumping to any conclusions and saying that this is the opinion for all programmers, it is a clear pattern with these test participants. With their frame of reference brought into the test session, it is my conviction that they think SpaceLab is easy to use because they recognise the interaction model from other programming software. This view is shared with Artman (1999) who mentions the information processing structures, which are learnt by interacting with the environment. Like Eysenck and Keane (2000) describe concerning pattern recognition the programming experienced test participants recognise patterns in the SpaceLab interaction. I think the fact that the programmers recognise themselves also matters in how easy they think it is to get lost, since none of them thinks it would be easy to do. Corresponding opinion for non-programmers are that it actually is quite easy to get lost within SpaceLab. This also confirms the recognition importance, since it naturally is easier to get lost in a place where a user has never been.

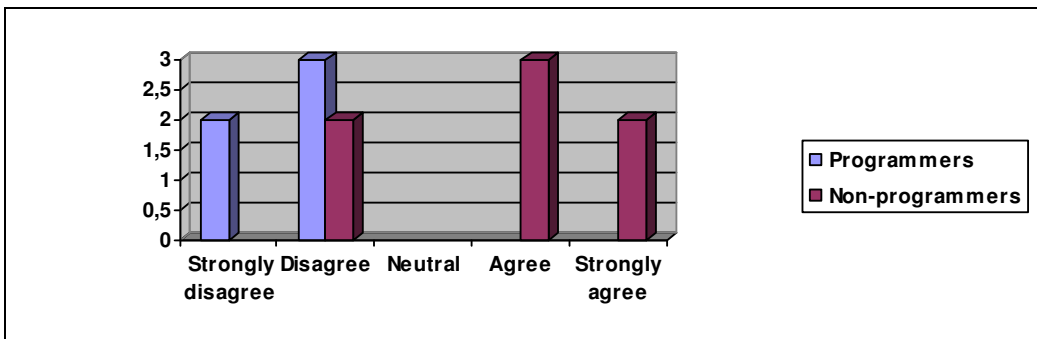
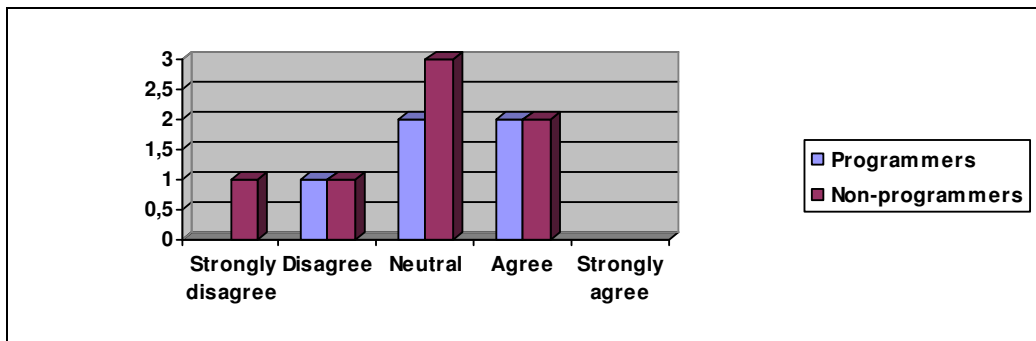


Figure 22. Level of agreement: It is easy to get lost in SpaceLab

Another question with answers that differs slightly depending on programming background is if the feedback within SpaceLab is sufficient. The programmers seem to be more satisfied with the amount of feedback presented than the non programmers, which could be explained by an uncertainty in the non programmers' interaction with the program. They are not sure what their actions will have for consequences and are therefore wishing for more and clearer feedback. Preece et al. (2002) says that it is fundamental for users to understand what their actions will have, and that good feedback really facilitates interaction with a system.



**Figure 23.** Level of agreement: Feedback is sufficient in SpaceLab

Most users thought that the help section was usable despite the fact that almost all of them had trouble when interacting with it. This part of the software could be hard to apply major changes to, since it is built on Windows standards and follow their principles. What must and can be improved is what terms that are included in the index, a major increase of key words are needed.

The fact that all participants, both those with programming background and those without, considered training a necessity to understand SpaceLab confirms the notion that it is a complex software. It is common sense that any new action would need training to be performed in an ultimate way, but a goal in software development must be to reduce the training needed by making the interface more intuitive. I think that the rather positive judgement SpaceLab received in the post test questionnaire is an overall sign that users think it is alright to require training to understand software. Note that I am not drawing any general conclusions, but only discussing the results of these tests and letting each reader draw their own conclusion. But a fact is that these test participants, members of our society, who all recorded at least eighteen evidence of usability problems during the approximately hour long session, think that SpaceLab is quite easy to use, if they could only receive some training first. I say that it should be possible to consider SpaceLab easy to use without considerable training. By applying the recommendations found in section 4.2 I am convinced that SpaceLab would be regarded as usable software by new users.

### **4.1.3 Further remarks**

Since this thesis are aiming at find usability problems in SpaceLab and reach an understanding about what new users find hard within the program, the recommendations in next chapter will be formulated from that point of view. But as SpaceLab are part of an existing toolkit, SpatialAce, it is of utmost importance that the possible consequences for the existing users are evaluated before any design alterations are made. These power users are customers that have learnt SpaceLab despite its flaws, and their opinions and habits should not be ignored.

When working with the development of an application it is important to strive for a pleasurable product, as promoted by Jordan (2002). And to achieve a pleasurable product the application must go through the stages of functionality and usability. In my opinion, SpaceLab still is in its functionality phase. With this thesis I hope to help it take the step towards a product that is both functional and usable as described by Norlin and Sjöberg (2002). The ways I hope to accomplish this is by lowering the learning threshold and give the users a feeling of adequacy earlier than it is given today. By lowering that threshold I think that the users can escape frustration at an early point in the learning process, and also reach the point where they feel passionate about the software at an earlier stage. This process is shown in a chart that does not really fit into the report, but explains clearly what I have tried to do in this thesis (Chart found in Appendix 6).

One way of lowering the learning threshold would be to act upon what Akoumianakis et al. (1997) wrote about user adapted interfaces. By disabling some parts of the program for users that register themselves as novices, I think the speed of the learning process would increase significantly. But the problem with such a solution would involve the choice of what functions that would be disabled, since it is hard to know what each user wants to use the application for. Therefore I believe that changing SpaceLab to a user adapted interface will cause more agony and frustration than pleasurable feelings of fast learning.

## **4.2 Recommendations**

The following recommendations for a design alteration of SpaceLab are a summary of the findings from reviewing of state-of-the-art theories and the results from the empirical study and might therefore be found in other parts of the thesis as well. They are not presented in any special order, meaning that they are not ranked among themselves. Some of the recommendations are similar to guidelines that other researchers have made, but in those cases I have tried to give an example of what I think would be beneficial to change in SpaceLab in relation to that guideline.

### **4.2.1 Design recommendations**

With regard to the research questions one of the most important recommendations must be to eliminate all bugs within the interface, in particular the problem when trying to preview a configuration in SpaceExplorer.

In order to facilitate for new users the introduction to the system must get better. I recommend an extensive tutorial that is offered to the user the first time SpaceLab is started, and also available from a menu. This tutorial should include an introduction to the system with clear examples of how the interaction works, along with exercises to give the user a possibility to try using the system with assistance. When the user has gone through the tutorial, I think that a box with tip-of-the-day should replace the tutorial at start-up. This box of tips must be possible to turn off, to avoid user annoyance.

Include a set of basic configuration files in the installation pack, containing the most common nodes and objects. Alternatively, create a wizard where you can choose all nodes and objects that should be included in your configuration file. Each node or object should be able to be chosen by a checkbox and automatically inserted in the configuration file when the confirmation button is pushed.

Make sure that the program is consistent and follows accepted standards. For instance, change the feedback for an incomplete node – red for conflict and yellow for incomplete is appropriate. Make feedback more visible, the red box with a white cross that marks an attribute row with a problem is too small. Try to use the same dialogues for similar choices, for instance the selection of colours; otherwise the user's mental models will not agree with the system model. Make sure that lists have similar appearances; if one list is sorted alphabetically then all of them should be sorted.

Allow the users to tailor their workspace, for instance by giving them a chance of adding and removing objects in the folder Common.

The SpaceLab user's guide would benefit from a review, making sure that everything is up to date and delete or rephrase confusing passages. A filtering function within the help section that can filter out all entries that not are related to Spacelab would shorten search times. I also think that new users would appreciate more tool tips when they hold the pointer over an object or a name, instead of having to check the help section or selecting the object. This feature must, as the tip-of-the-day box, include a possibility to turn it off since it can be annoying for users when they have learnt the program.

Make better use of the right click menu by including more functionality. For instance, SpaceLab would benefit from a possibility to change node sizes from the right click menu and an option to add objects that now only are found in the class library.

Another important issue is the error messages; they must be revised so that they only give the user information that he can benefit from. Avoid using code and make sure that they always include information about what was done wrong and how that problem can be avoided in the future.

The test results clearly show that new users have problems with the programming related terminology in SpaceLab. Trying to reduce this would enhance their possibility to learn the program fast, although this probably is impossible to implement due to SpaceLab's close relation to other software.

A small alteration in the visualisation of the nodes that would improve the overview of the configuration files is if the nodes were drawn in straight lines. If such a change would happen, the configuration files would probably grow slightly but I think that what is lost in size is more than recovered in the clear overview.

### **4.2.2. Overall recommendations**

This thesis will probably not lead to a total change in SpaceLab appearance, but only see to that the most obvious usability problems are attended to. But when such an (major) update are about to happen I think it is important to carefully study the new standards in the software market. SpaceLab today looks like typical Windows software, which naturally has both positive and negative sides. It can be assumed that most users run SpaceLab at a Windows based system, giving them a feeling of recognition since it looks like other Windows programs. But as Microsoft is on the verge of releasing their new Windows Vista, it can be wise to look at the new features included, since those will probably set new standards for what users expect in an application.

If a major redesign of the appearance of SpaceLab is carried out I would recommend that a review of the working process is done prior to this redesign. Suggestions in how the process can be changed can be found in the ISO standards for usability. By involving an interaction designer in the project, and thus introduce users at an early stage in the development process I am convinced that the final result will be more user friendly than it would if these people are left out. I think it is important that the thought design is tested with potential users at an early stage, in order to have time to make corrections before the project is too far into the development process.

### **4.3 Conclusion**

The main purpose of this thesis was to find usability problems in the visual tool SpaceLab and propose design alterations to abolish these flaws. I think that I have managed to accomplish the undertaken assignment by thoroughly evaluating the topical software, by performing an empirical study and by obtaining results. These results constituted, together with relevant theoretical guidelines, the base of material that helped me to answer the research questions. The methods used worked well and proven by the quality of the gathered material it can be said that correct methods were chosen.

The few problems that arose during the research process were dealt with accordingly and did not affect the result in a negative way. Instead I feel that the minor setbacks experienced have strengthened me as a researcher and given me possibilities to practice my abilities in problem solving.

#### ***4.4 Future research***

There is a vast array of possible research areas relating to the performed study. A study that is directly connected to the results of this thesis would look into a development of the recommendations made. It could include suggestions of how the recommendations could be included in SpaceLab and a performance of user tests to optimize these improvements.

Later would an evaluation of the outcome of the implemented recommendations be interesting to see. This could be done simultaneously as an evaluation of the more advanced features in SpaceLab, concentrating on power user's interaction models and habits.

It would also be interesting to see how SpaceLab is used by customers at their normal working location. An on-site user evaluation studying interaction patterns and features used would be of great use to Carmenta. Such a study could include usability tests, interviews and questionnaires.

Another way of collecting feedback concerning SpaceLab and SpatialAce would be by sending a mail to customers including a link to a questionnaire, or by simply include a questionnaire into the SpaceLab program that opens up after the program being started a predefined number of times. The questionnaire must be optional to answer if the second choice is used, otherwise you might offend perfectly good customers.

---

## REFERENCES

- ACM SIGCHI. (1992). *Curricula for Human-Computer Interaction*. [WWW-document]. URL: <http://sigchi.org/cdg/cdg2.html>. (2005-09-13).
- Akoumianakis, D., Paramythis, A., Savidis, A. & Stephanidis, C. (1997). Designing user-adapted interfaces: the unified design method for transformable interactions. In *Proceedings of the Conference on DIS'97*. Amsterdam, Netherlands.
- Artman, H. (1999). *Fördelade kunskapsprocesser I ledningscentraler vid nödsituationer - koordination och situationsmedvetenhet*. Linköping: Institutionen för Tema Kommunikation, Linköpings Universitet.
- Backman, J. (1998). *Rapporter och uppsatser*. Lund: Studentlitteratur.
- Barnum, C. (2002). *Usability Testing and Research*. United States of America: Pearson Education Inc.
- Bevan, N., Kirakowski, J. & Maissel, J. (1991). What is Usability? In *Proceedings of the 4th International Conference on HCI*. Stuttgart, Germany.
- Broadbent, D. E. (1958). *Perception and communication*. Oxford. Pergamon.
- Coren, S., Ward, L. M. & Enns, J. T. (1999). *Sensation and Perception*. (5<sup>th</sup> ed). Orlando: Harcourt Brace.
- Dicks, R. S. (2002). Mis-Usability: On the Uses and Misuses of Usability Testing. In *SIGDOC'02*. (pp. 26-30). Ontario, Canada.
- Dodwell, P. C. (1994). Fundamental processes in vision. In: R. L.Gregory & A. M. Colman (Eds.), *Sensation and perception* (pp. 1- 23). London: Longman.
- Dumas, J. S. & Redish, J. C. (1999). *A Practical Guide To Usability Testing*. Intellect, Ltd: Bristol.
- Ebling, M. R. & John, B. E. (2000). On the Contributions of Different Empirical Data in Usability Testing. In *Proceedings of the conference on DIS'00*. New York, USA.
- Eriksen, C. W. & St. James, J. D. (1986). Visual attention within and around the field of focal attention: A zoom lens model. In *Perception & Psychophysics*, 40. 225 – 240.
- Ericsson, K. A. & Simon, H. A. (1980). *Psychological Review*. 87, 215-251.
- Ericsson, K. A. & Simon, H. A. (1984). *Protocol Analysis: Verbal Reports as Data*. Cambridge: MIT Press.
- Eysenck, M.W & Keane, M.T. (2000). *Cognitive psychology - A Student's Handbook*. (4<sup>th</sup> ed.) Hove: Psychology Press, Publishers.

## REFERENCES

---

- Faulkner, C. (1998). *The essence of human computer interaction*. Essex: Prentice Hall.
- Faulkner, L. (2003). Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. In *Behavior Research Methods, Instruments & Computers*. 2003, 35 (3), 379-383.
- Gulliksen, J. & Göransson, B. (2002). *Användarcentrerad systemdesign: en process med fokus på användare och användbarhet*. Lund: Studentlitteratur.
- Holleran, P. A. (1991). A methodological note on pitfalls in usability testing. In *Behaviour & Information Technology*. 10, (5), 345-357.
- Holme, I. M. & Solvang, B. (1997). *Forskningsmetodik – Om kvalitativa och kvantitativa studier*. (B. Nilsson, Trans.) Lund: Studentlitteratur. (Original work published 1996).
- Hartman, J. (1998). *Vetenskapligt tänkande - från kunskapsteori till metodteori*. Lund: Studentlitteratur.
- Hertzler, E. G. & Nowell, L. T. (1998). Graphical Encodings: Bet You Can't Use Just One! In *Proceedings of NPIV '98*. (pp. 3-8). Washington, DC.
- ISO 9241:11 (1998). *Guidance on Usability*. [WWW-document]. URL: [http://www.hostserver150.com/usabilit/tools/r\\_international.htm#9241-11](http://www.hostserver150.com/usabilit/tools/r_international.htm#9241-11). (2005-09-13).
- James, W. (1890). *Principles of psychology*. New York: Holt.
- Jordan, P. W. (2002). *Designing Pleasurable Products – An introduction to the new human factors*. New York: Taylor & Francis.
- Kantner, L. (1994). Techniques for Managing a Usability Test. In *IEEE Transactions on Professional Communication*, 37, (3), 143-148.
- Kerlinger, F.N. (1979). *Behavioral Research*. New York: Holt, Rinehart & Winston.
- Kvale, S. (1997). *Den kvalitativa forskningsintervjun*. (S-E Torshell, Trans.) Lund: Studentlitteratur. (Original work published 1996).
- Lundh, L-G., Montgomery, H. & Waern, Y. (1992). *Kognitiv psykologi*. Lund: Studentlitteratur.
- Meier, B. J. (1988). ACE: A Color Expert System for User Interface Design. In *Proceedings of ACM SIGGRAPH 1988*. (pp. 117-128). Alberta, Canada.
- Nielsen, J. (1990). *How to Conduct a Heuristic Evaluation..* [WWW-document]. URL: [http://www.useit.com/papers/heuristic/heuristic\\_evaluation.html](http://www.useit.com/papers/heuristic/heuristic_evaluation.html) (2005-09-11).

## REFERENCES

---

- Nielsen, J. (1992). Finding Usability Problems Through Heuristic Evaluation. In *Proceedings of ACM CHI'92 Conference*. (pp. 373-380).
- Nielsen, J. (1993). *Usability Engineering*. London: Academic Press
- Nielsen, J. & Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of ACM CHI'90 Conference*. (pp. 373-380). Monterey, WA.
- Norlin, C. & Sjöberg, A. S. (2002). Wanted: A Wider Scope for Interaction Design in Sweden. In *NordiCHI'02*. (pp. 235-237). Århus, Denmark.
- Norman, D. (1988). *The Design of Everyday Things*. New York: Doubleday.
- Pervin, L.A. (1984). *Personality*. New York: Wiley.
- Preece, J., Sharp, H. & Rogers, Y. (2002). *Interaction Design – Beyond Human-Computer Interaction*. New York: Wiley.
- Proctor, R. W. & van Zandt, T. (1994). *Human factors in simple and complex systems*. Needham Heights: Allyn & Bacon.
- Roth, I. (1986). An introduction to object perception. In Roth, I. & Frisby, J. P. (Eds.) *Perception and representation: A cognitive approach*. Milton Keynes: Open University Press.
- Russo, J., Johnson, E. & Stephens, D. (1989). The validity of verbal protocols. *Memory and Cognition*, 17, 759-769.
- Rubin, J. (1994). *Handbook of Usability Testing*. New York: Wiley.
- Tufte, E. (1990). *Envisioning Information*, Cheshire: Graphics Press LLC.
- Wichansky, A. M. (2000). Usability testing in 2000 and beyond. In *Ergonomics*, 43, (7), 998-1006.
- Winograd, T. (1997). From Computing Machinery to Interaction Design. In Denning, P. & Metcalfe, R. (Eds.) (1998). *Beyond Calculation: The Next Fifty Years of Computing*. New York: Springer.

## Appendix 1 - Usability Test Planning

### Test team:

Test leader/Experimenter: Johan Ågren  
Observer: Tobias Moberg  
Possible observers: Other Carmenta employees

### Test date details:

Test planning/construction: 5/10 – 28/10 according to project scheme.  
Performing test: 1/11 – 18/11 according to project scheme.  
Structure and Analyse test results: 17/11 – 9/12 according to project scheme.

### Test objectives:

Finding and documenting usability problems within SpaceLab, as they are found by users new to the system.

### Test details:

Location: Carmenta office in Gothenburg.  
Test software: SpaceLab, running at a normal state.  
Recording: Software called Morae.  
Documents: Participants must sign consent form before test starts. Participants are also asked to fill out post test questionnaire concerning their impressions of SpaceLab.  
Time: Each session are expected to last maximum two hours, including instructions and post test questionnaire.

### Observer details:

The observer(s) will be located in a different room from where the test participants are conducting the tests, in order to make the user feel more comfortable. These way observers can take notes or comment on the test results without running the risk of interfering with the test process.

### Number of test users:

Ten. As recommended by different usability gurus (See section about choosing participants), a number that is greater than five is used. This way the researcher hope to locate a larger number of usability problems than a five-participant test would have found.

### User characteristics:

The participants that are included in the test should not have any earlier experience from working with the system.

Five of the participants should have programming background and should be familiar with the way that for instance Visual Basic is constructed (meaning that they know the principle of changing properties of a selected object within its object editor).

Five of the participants should have no or very little programming experience. This is a deliberate choice in order to see if programming background has any influence on how the user perceives the system.

**User introduction prior to test:**

The users are given an introduction to the system prior to the test, in order to give them enough knowledge about SpaceLab to have possibilities to conduct the test. Instructions provided in appendix 3.

**Help available to test participants:**

During the test the participants have access to the help functions within SpaceLab and may use them in whatever way they feel.

**Data collection:**

During the tests data collection will be continuous, saving sounds from the microphone, videos from the camera, keyboard actions, screen caps, mouse clicks and application events to facilitate a thorough analysis in later stages of the thesis work.

**Test tasks:**

The tasks that the users will perform during the test are displayed in appendix 3.

**Usability problem definition:**

To ease the analysis process a definition of what constitutes a usability problem had to be formulated. In this test as suggested by Ebling and John (2000) a usability problem is identified if:

- *“two or more sources of evidence suggest a problem,*
- *two or more users suggest a problem,*
- *one or more users crash the interface,*
- *one or more users identify a bug in the interface, or*
- *one user’s evidence suggests a problem and the author concur.”*

(Ebling and John, 2000)

## Appendix 2 - Heuristic Evaluation

### *SpaceLab*

#### *Instruktioner*

Du kommer nu att utföra en Heuristic Evaluation på en del av utvecklingsmiljön SpatialAce, utvecklat av Carmenta. Utvärderingen kommer att inledas med att du får möjlighet att bekanta dig med systemet under en valfri tidsperiod (dock ej längre än en timme). Under denna tid kommer testledaren att vara tillgänglig för frågor som handlar om användandet av systemet. Efter den initiala kontakten med systemet kommer testledaren att förklara det set av heuristics som kommer att användas under testet. Dessa finns även skriftligt att tillgå under hela utvärderingen.

Utvärderingen utförs enligt ett förutbestämt program där ett inledande scenario med uppgifter ska utföras av dig, när dessa uppgifter är avklarade får du fritt undersöka programmet och hitta fler fel i användbarheten.

För att underlätta efterarbetet och förbättra förutsättningarna att komma ihåg de användbarhetsproblem som du hittar, är det en önskan att du under hela testet antecknar de användbarhetsproblem som du hittar. Dessa anteckningar ska refereras till vilken heuristic som det aktuella problemet bryter mot.

Lycka till!

Scenario:

Du ska skapa en världskarta som när den är färdig är möjlig att granska i SpaceExplorer.

Följ stegen nedan (engelska, test 1 finns även i User's guide benämnt Hello World)

#### **Test 1**

*Creating a simple configuration and reviewing it in SpaceExplorer.*

**Step 1.** Start SpaceLab. Create an empty configuration. Choose the New command on the File menu or on the toolbar. SpaceLab will now create a new configuration, most likely with the default name Map0, and a dataflow window will appear on the desktop panel.

**Step 2.** Create a View node. Make sure the class index is visible, and select the Common folder in the class tree or the Views node in the SpatialAce folder. A list of classes will appear in the list to the right of the tree, and one of them will have a pink box icon and be named View. Drag this class to the dataflow window and drop it on

the background. You should get a green box (with a pink icon) labelled View0. Now, you should set the reference system of the View. Locate RefSys.WGS84lola in Resources index, drop it on View0 and see how the refSys property of View0 is changed. Another way to set the reference system of the View is to choose a one using the drop down box of the refSys property in the object editor area below the Class index.

**Step 3.** Create a Layer. An OrdinaryLayer will do fine. This is also one of the common classes. Locate the class, drag it to the dataflow window, and drop it on top of the View0 object. You should get another green box, this time labelled Layer0, to the right of the View0 object.

**Step 4.** Create a RenderOp. This is where the land contours will get their color. Locate the RenderOp class and drop it on the Layer0 object. You may have noticed that the Layer0 box turned gray, which is because it no longer has any undefined properties.

**Step 5.** Create a ReadOp. This is the data source. The ReadOp class is listed among the common classes. Find it and drop it on the RenderOp0 object. Two of the boxes are still green, however, which means that they need attention.

**Step 6.** Add a Window to the View. The View class has a required property called drawable, which is where the map is drawn. A Window object is suitable in our case. Find the Window class and drag it to the View0 object. An item called Window0 will appear below the View0 object. Now, you should set the background color of the window. The Window0 object is selected and displayed in the object properties editor. The default value of background is black, [0, 0, 0]. Select the background value and invoke the custom color editor by pushing the button ... at the end of the value field, or by a double-click on the background row, or by choosing "Edit in custom editor" from the right mouse button popup, or by pressing CTRL-E when the background row is selected. Find a nice sea-blue color and press OK.

**Step 7.** Add a PolygonVis to the RenderOp. We would like the land contours to be painted green on top of the blue background. Locate the PolygonVis class and drag it to the RenderOp. Pick the visList item from the popup menu that appears. In the object properties editor, edit the color like in step 6, but make it forest green this time and use the text editor by pushing the button T at the end of the value field or by pressing CTRL-T.

**Step 8.** Add a DataSet to the ReadOp. We will use the Digital Chart of the World from NIMA. Find the VPF class: browse the list of classes you get after selecting the DataSet node in the SpatialAce folder. Drag it to the ReadOp. Set the databasePath to Local.DcwHome (either by dragging the system resource, by typing or by choosing from the drop-down list), set libraryName to "BROWSE", set coverageName to "PO", and set featureClass to "POAREA".

**Step 9.** Add a DataSetQuery to the ReadOp. Find the VPFQuery class under the DataSetQueries node. Drag it to the ReadOp. Set the typeColumn property to

"POPYTYPE" and set keepList to [ 1 ]. This means we are interested in polygons with type code = 1, which are the land contours.

**Step 10.** Send the configuration to SpaceExplorer. This is really easy, just push the SpaceExplorer button (the little globe icon) on the toolbar or press F5, and things will happen.

### **Test 2**

*Apply a change within an existing configuration.*

Open a configuration file called dcw.p that can be found in the catalog  
C:\program files\carmenta\spatialace4\samples\spacelab\

You are now supposed to alter the configuration by adding a different filter into the Layer called "Countries"

**Step 1.** Check the configuration file in SpaceExplorer as described above. Then Close SpaceExplorer. Delete the query: overviewLandQuery in the Layer Countries. Save file as "dcw05.p".

**Step 2.** Check the file in SpaceExplorer again. Note the difference. Close SpaceExplorer.

**Step 3.** You are now going to add a filter operator that will make sure that the polygons who represent sea is not visualized. Add a AttFilterOp between the RectClipOp and the ReadOp in Countries.

**Step 4.** Define an andCondition in AttFilterOp. It should have a name - POPYTYPE and a value - 1. Save file.

**Step 5.** Check the configuration in SpaceExplorer. It should now look the same as the first time you checked it.

End of test 2! Now explore SpaceLab on your own, don't forget to note any usability flaw you find and make a reference to correct heuristic that it concerns.

## Appendix 3 - Testmanus

### ***Introduktion för användaren.***

[Välkomna deltagaren och erbjud de förfriskningar som finns att tillgå. Visa var toaletten finns.]

Jag vill börja med att tacka för att du har kommit hit idag och gör det möjligt för oss att användbarhetstesta den här produkten vi ska titta på idag.

Jag heter Johan och kommer att leda dagens test.

Vi jobbar med att förbättra användbarheten med systemet SpaceLab och som en del av det arbetet ber vi ett antal människor försöka utföra en mängd olika uppgifter för att se vilka delar i systemet som kan orsaka problem. Det är alltså systemet som testas och inte din skicklighet att utföra uppgifterna. Om du känner att det är delar av SpaceLab som är svåra att förstå, så är risken stor att det även är andra som känner så och det är då vårt jobb att förbättra dessa områden.

För att vi ska kunna analysera den information som testet ger oss lite djupare skulle vi vilja videofilma användbarhetstestet, så om du läser och skriver på den här tillåtelsen så kan vi snart komma igång. [Ge användaren blanketten med medgivandet]

Under testet vill vi att du använder dig av ett så kallat tänka högt protokoll. Det vill säga att du under hela testet talar om vad du tänker och vad du gör. [Demonstrera tänka högt protokoll] Om du skulle glömma bort att tala om hur du tänker, vilket är lätt hänt så kommer jag att påminner dig, eller ställa en fråga om hur du tänker vid en viss situation.

När du genomför testet kommer andra personer att observera och dom kommer jag att introducera dig till nu. [Presentera ev. observatörer]

Dagens test kommer att pågå under ca 1,5 timme. Vill du ta en paus under testet så säg bara till.

Jag vill också tillägga att om du känner att du av någon anledning inte vill fullfölja testet så får du avbryta när du vill.

[Ta med användaren till testrummet]

## **Introduktion till systemet**

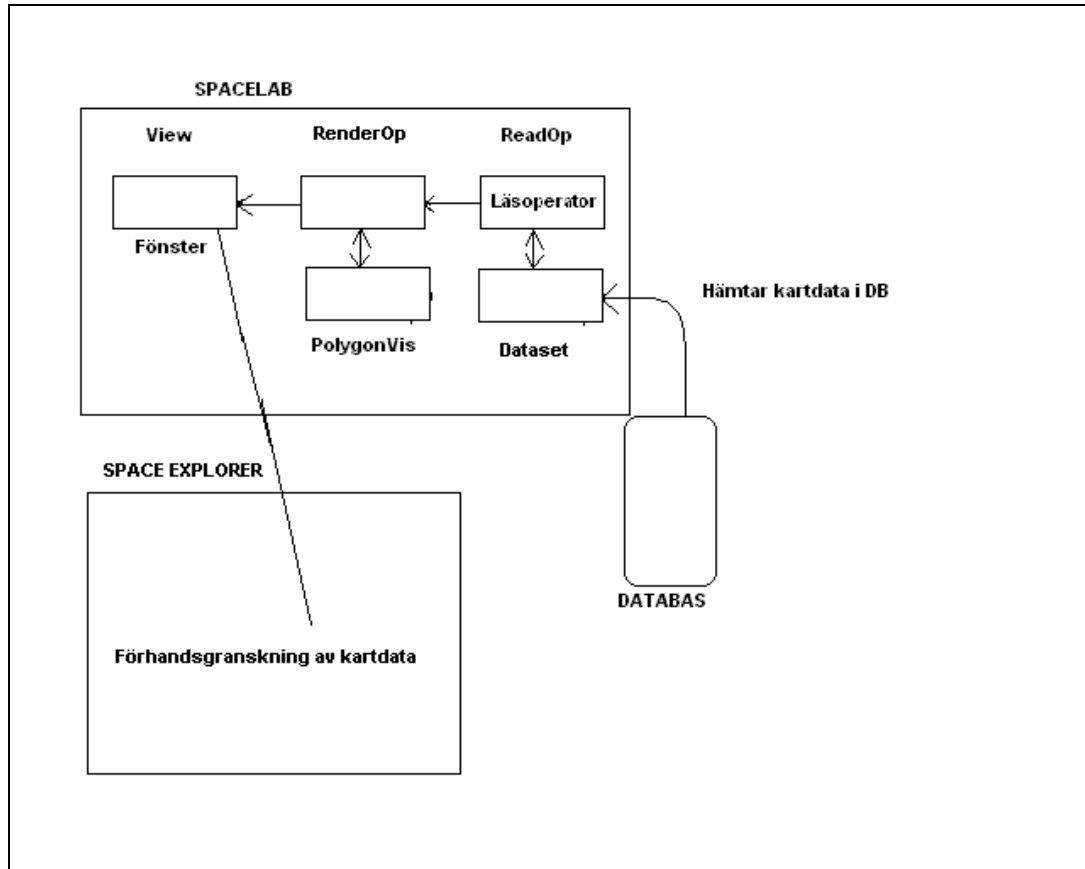
Jag kommer nu att hålla en kort introduktion för dig om SpatialAce och SpaceLab i synnerhet, så att du ska veta lite hur systemet fungerar när vi börjar med uppgifterna sedan.

SpatialAce är en produkt som Carmenta utvecklat under lång tid, den har främst inriktat sig till programutvecklare men på senare tid har man sett en möjlighet att öka intressentsfären. Själva SpatialAce är en verktygslåda med ett antal olika verktyg för visualisering av olika kartdata. Dessa kartdata organiseras i konfigurationsfiler kallade .p i utvecklingsmiljön SpaceLab.

## **SpaceLab**

[Ge användaren ett exemplar av bilden nedan]. Jag ska försöka förklara den här bilden nu, om du har några frågor är det bara att ställa dom. SpaceLab som är den del av SpatialAce som vi ska undersöka närmare idag är uppbyggt av operatörer som gör utför olika uppgifter, och de här operatörerna delas in i olika klasser. Konfigurationsfilerna skapas genom att operatörer väljs i klassbiblioteket och sedan dras ut till dataflödesfönstret. För att sedan lägga till objekt till en operator släpps objektet helt enkelt ovanpå den önskade operatören. Ett exempel med operatörer är att alla konfigurationsfiler har en läsoperatör kallad ReadOp som läser in kartdata i ett dataset från en databas. Till denna läsoperatör läggs sedan andra operatörer, till exempel en visualiseringsoperatör kallad RenderOp. RenderOp tar hjälp av en PolygonVis för att visualisera polygoner. Det finns med andra ord olika visualiserare för olika objekt, till exempel textVis för text osv. Efter att ReadOp läst in kartdatan från datasetet talar PolygonVis om hur den ska ritas ut. I motsatt ände av kedjan av olika operatörer finns alltid ett fönster, en View, där kartdatan ska ritas ut utefter de egenskaper som de olika operatörerna satt upp. Används många olika dataset så är det vanligt att man sorterar in kedjor av operatörer i olika lager, för att få en bättre överblick på trädstrukturen.

SpaceLab bygger just på ett sådant trädstrukturerat flödestänkande, information hämtas från en databas, sedan bestämmer olika operatörer hur den ska presenteras i det fönster som ofta integreras i ett program. Alla .p-filer kan förhandsgranskas i ett program som heter SpaceExplorer.



Det här är alltså ett exempel på en beskrivning av upplägget i SpaceLab. Hur kartan ser ut i SpaceExplorer beror på vilka data som läses in, vilka operatörer som finns med i kedjan och vilka egenskaper användaren ger dom. Om du vänder på bladet har du en översikt över hur gränssnittet ser ut, med klassindex överst till vänster, under det ett fält med egenskaper för de olika objekten och till höger det så kallade dataflödesfönstret.

Har du några direkta frågor innan vi sätter igång?

### ***Förklaring av hjälpmedel***

[Förklara vilka hjälpmedel som används under användbarhetstestet]

Vi kommer att använda oss av ett loggningsprogram som heter Morae för att dokumentera det här testet, det spelar in hur du navigerar på skärmen och samtidigt hur du reagerar på feedback från systemet.

### ***Testet***

Vi har totalt 8 antal uppgifter. Jag kommer att ge dig dessa en och en, och dessutom läsa upp dom för dig när du får dom. Kom ihåg att tänka högt när du försöker lösa

dom. Jag vill dessutom igen poängtera att det är SpaceLab som vi utvärderar och inte dina färdigheter.

Har du några frågor innan vi börjar??

Då startar vi inspelningen. [Starta kamera/program]

## Uppgift 1

[Läs uppgift 1 och ge användaren ett exemplar av densamma]

### Scenario 1.

**Tid:** 5 min

**Scenariotext:** Du har precis laddat ner en provlicens av ett program som du hört att det ska gå att göra kartor med och öppnar programmet för första gången. Du vill därför navigera dig runt i programmet och skapa dig en första uppfattning om det. Berätta högt om dina första intryck av SpaceLab. Öppna gärna en konfigurationsfil om du vill det, dom finns lagrade i mappen som du befinner dig i när du trycker på öppna.

[När användaren är klar] Tack, då fortsätter vi till nästa uppgift.

## Uppgift n

[Presentera alla uppgifterna på samma sätt och i rätt ordning.]

### Scenario 2.

**Tid:** 5 min

**Scenariotext:** Du vet nu ungefär hur interaktionen med SpaceLab fungerar, och vilka beståndsdelar som finns. En kompis till dig som inte fått någon introduktion till systemet har svårare att förstå vad som menas med uttrycket Classes, så han kommer till dig och ber om hjälp. Hitta ett sätt i programmet som hjälper dig att förklara vad det innebär och berätta vad du hittar.

**Mål:** Kontrollera hur enkelt användaren hittar till ett visst hjälpavsnitt.

**Måtkriterier:** Användaren hittar till hjälpavsnittet och förklarar begreppet Classes inom 5 minuter.

### Lista uppgifter:

- Hitta hjälpen
- Öppna hjälpfunktionen

- Öppna SpaceLab 2 User's guide
- Klicka på Classes
- Läs texten

Eller

- Sök efter class i fliken Search
- Välj SpaceLab 2 User's guide
- Välj Display
- Klicka på Classes
- Läs texten

### Scenario 3.

**Tid:** 10 min.

**Scenariotext:** För att förstå sambandet mellan SpaceLab och SpaceExplorer (där konfigurationsfilerna förhandsgranskas) vill du kontrollera vad som händer med en kartas utseende när du förändrar ett attribut i SpaceLab. Öppna kartan dcw som ligger i mappen C:\program files\carmenta\spatialace4\samples\spacelab\. Kontrollera hur den ser ut i SpaceExplorer och stäng sedan programmet. Ändra färg på länder och sjöar/hav i konfigurationsfilen i SpaceLab och kontrollera sedan resultatet i SpaceExplorer.

**Mål:** Kontrollera hur lätt det är för användaren att lokalisera det attribut som behöver ändras. Kontrollera användbarheten i dialogen för färgval. Kontrollera om användaren förstår att kartan måste sparas innan den kan förhandsgranskas i Explorer. Se om användaren förstår hur en karta förhandsgranskas.

**Måtkriterier:** Användaren lyckas ändra färg på länder och hav i kartan dcw.p

#### Lista uppgifter:

- Öppna kartan dcw.p
- Lokalisera lagret countries och attributet Landcolor
- Ändra färg på Landcolor
- Lokalisera lagret lakes och attributet Watercolor
- Ändra färg på Watercolor
- Spara kartan
- Förhandsgranska i SpaceExplorer

### Scenario 4a.

**Tid:** 5 min.

**Scenariotext:** Du har fått en felaktig konfigurationsfil skickad till dig av en kompis som inte kan lokalisera var problemet är. Den heter kartatrasig.p och ligger i samma

mall där du tidigare öppnade en karta. Försök hitta var felet är och exakt vad som är fel. Berätta högt om vad du hittar och hur du hittade det.

**Mål:** Se hur användaren lokaliserar en ofullständig nod och på vilket sätt han upptäcker vad som är felet. Kontrollera om användaren tror att markören för en kommenterad nod har något med felet att göra.

**Måtkriterier:** Användaren lokaliserar och talar om vilken nod det är fel på, samt vad felet består av dvs. att ReadOp saknar ett dataset.

**Lista uppgifter:**

- Öppna kartan kartatrasig.p
- Lokalisera den ofullständiga noden
- Jämför noden med övriga ReadOp
- Hitta det som saknas i jämförelsen

**Scenario 4b.**

**Tid:** 10 min.

**Scenariotext:** Du har kommit fram till att konfigurationsfilen saknar ett dataset i lagret countries. Datasetet som ska användas heter VPF och finns i klassbiblioteket. För att kunna se kartan måste ett sådant dataset läggas till och ges attributen coverageName: "PO", featureClass: "POAREA" och libraryname: "BROWSE". Sökvägen till databasen ska anges till densamma som dataseten i de andra lagren har. Gör de ändringar i konfigurationsfilen som behövs för att kunna förhandsgranska den.

**Mål:** Kontrollera om användaren kan åtgärda felet med hjälp av korta instruktioner i uppgiften. Kontrollera om inmatningssätten med "" runt t.ex. databasePath är självklart.

**Måtkriterier:** Användaren lokaliserar datasetet VPF, placerar det på rätt plats och ger det rätt attribut. Användaren lyckas förhandsgranska kartan.

**Lista uppgifter:**

- Lokalisera datasetet VPF i SpatialAce Datasets
- Dra datasetet till ReadOp
- Sätt attributet coverageName till "PO"
- Sätt attributet featureClass till "POAREA"
- Sätt attributet libraryname till "BROWSE"
- Sätt sökvägen databasePath till Local.DcwHome
- Spara konfigurationsfilen
- Förhandsgranska konfigurationsfilen

**Scenario 5.**

**Tid:** 10 min.

**Scenariotext:** Ett vanligt sätt att visualisera objekt i en karta är med hjälp av så kallad attributstyrd visualisering. Det vill säga att data som läses in från ett dataset ritas upp olika beroende på vilket nyckelvärde de har. Exempel är för städer, där storleken på staden på kartan kan bero på folkmängden i staden där exempelvis Alingsås som är mindre än Göteborg får ett lägre värde, och därför ritas upp mindre på kartan.

Samma princip används för att rita ut vägar. I kartan scenario5 (som du hittar i samma katalog där övriga kartor funnits) har någon missuppfattat hur nyckelvärdena fungerar, vilket gör att vägkanterna är alldeles för breda och dom små vägarna ser större ut än de stora. Ändra i värdena så att det syns på kartan vilka vägar som är stora och vilka som är små.

**Mål:** Kontrollera att användaren förstår vad som menas med attributstyrd visualisering. Kontrollera att användaren förstår dialogen som visas. Kontrollera om det lätt att hitta var nyckelvärdena finns.

**Måtkriterier:** Användaren hittar visualiseraren och lyckas öppna upp dialogen för hur breda vägkanterna är. Där ska användaren ha lyckats ändra värdena och se skillnaden i kartan som förhandsgranskas.

**Lista uppgifter:**

- Öppna karta scenario5.p
- Förhandsgranska i SpaceExplorer
- Stäng SpaceExplorer
- Lokalisera visualiseraren RoadEdgevis
- Hitta attributet width
- Öppna dialogrutan för width
- Ändra värdena för hur breda vägkanterna ska vara
- Spara kartan
- Förhandsgranska

**Eventuellt Scenario 6.**

**Tid:** 15 min.

**Scenariotext:** Du känner att du vill prova att konstruera din första egna karta. I hjälpen (SpaceLab2 User's Guide, avsnitt 4) finns ett exempel som heter "Hello World" vars innehåll sammanfattas nedan. Ditt mål är att konstruera en karta, utifrån instruktionerna i den här uppgiften och om det behövs med hjälp av exemplet "Hello World".

Instruktioner:

Din karta ska bestå av en View, ett OrdinaryLayer, en RenderOp och en ReadOp. Noderna ska vara kopplade från vänster i nämnda ordning. I View ska du sedan lägga till ett objekt kallat window och ett referenssystem kallat RefSys.WGS84lola. Sätt bakgrundsfärgen i Window till en havsblå färg.

Lägg till en PolygonVis i RenderOp, välj alternativet visList och sätt färgattributet till en grön färg. Efter det är det dags att bestämma vilket dataset som ska läsas in till ReadOperatorn. Lägg till ett VPF dataset och sätt sökvägen till databasen till Local.DcwHome. Ge sen datasetet samma attributvärden som i uppgift 4b, dvs coverageName: "PO", featureClass: "POAREA" och libraryname: "BROWSE".

För att bara läsa ut landskonturer på din karta måste du lägga till en datasetförfrågan. Utrusta ReadOp med en datasetQuery kallad VPFQuery och ge den attributen typeColumn: "POPYTYPE" och keepList: [1]. Spara som [ditt namn]\_karta och förhandsgranska.

**Mål:** Se hur användaren bygger sin första egna konfiguration. Kontrollera om användaren anser att hjälpen är användbar och förståelig. Kontrollera om användaren förstått hur programmet är upplagt och interaktionen fungerar.

**Måtkriterier:** Användaren lyckas skapa en karta från en helt tom fil. Användaren tar hjälp av hjälpen när han fastnar.

**Lista uppgifter:**

- Skapa en ny konfigurationsfil
- Lokalisera noderna View, OrdinaryLayer, RenderOp och ReadOp och lägg till i konfigurationsfilen.
- Lägg till ett Window i View
- Ange RefSys till RefSys.WGS84lola
- Ange bakgrundsfärg i View till en blå nyans
- Lokalisera PolygonVis i Visualisers och lägg till i RenderOp.
- Ändra färgattributet hos PolygonVis till grönt
- Lägg till ett VPFdataset i ReadOp
- Sätt sökvägen till databasen till Local.DcwHome
- Sätt attributet coverageName till "PO"
- Sätt attributet featureClass till "POAREA"
- Sätt attributet libraryname till "BROWSE"
- Lägg till en VPFQuery i ReadOp
- Sätt attributet typeColumn till "POPYTYPE"
- Sätt attributet keepList till [1]
- Spara kartan som användarens namns karta
- Förhandsgranska i SpaceExplorer.

**Scenario 7.**

**Tid:** 5 min.

**Scenariotext:** Du vill undersöka en detalj i en konfigurationsfil och jämföra den med några andra filer. För att få en bättre överblick på flera konfigurationsfiler samtidigt vill du först minimera noderna i kartan. Öppna kartan dcw som ligger i mappen C:\program files\carmenta\spatialace4\samples\spacelab\ och gör minimera alla noder på ett så smidigt sätt som möjligt. Öppna sen två valfria kartor ur samma katalog och

minimera de noderna också, men gör dom bara så små att du ser vad noderna heter. Se sedan till att du ser alla tre kartor på samma gång.

**Mål:** Kontrollera om användaren uppfattar att han kan minimera alla noder i konfigureringsfilen med hjälp de ikoner som finns i menyraden. Kontrollera att användaren förstår hur han organiserar de olika kartfönstren.

**Måtkriterier:** Användaren lyckas minimera alla noder i en .p-fil på några få musklick. Användaren lyckas få de tre kartfönstren synliga på samma gång.

**Lista uppgifter:**

- Öppna en .p fil.
- Lokalisera minimeringsikonen i menyraden
- Minimera ikonerna
- Öppna två kartor
- Minska storleken på noderna i kartorna till hälften
- Organisera kartfönstren så alla tre kartor syns samtidigt

## **Enkät**

[När användaren är färdig med alla uppgifterna] Tack så mycket. Då stoppar vi inspelningen. [Stäng av kamera/program]

Jag har en kort enkät här som handlar om dina intryck av SpaceLab som jag skulle vilja att du fyller i. Den informationen är givetvis också konfidentiell och kommer bara att användas till att förbättra utformningen av SpaceLab.

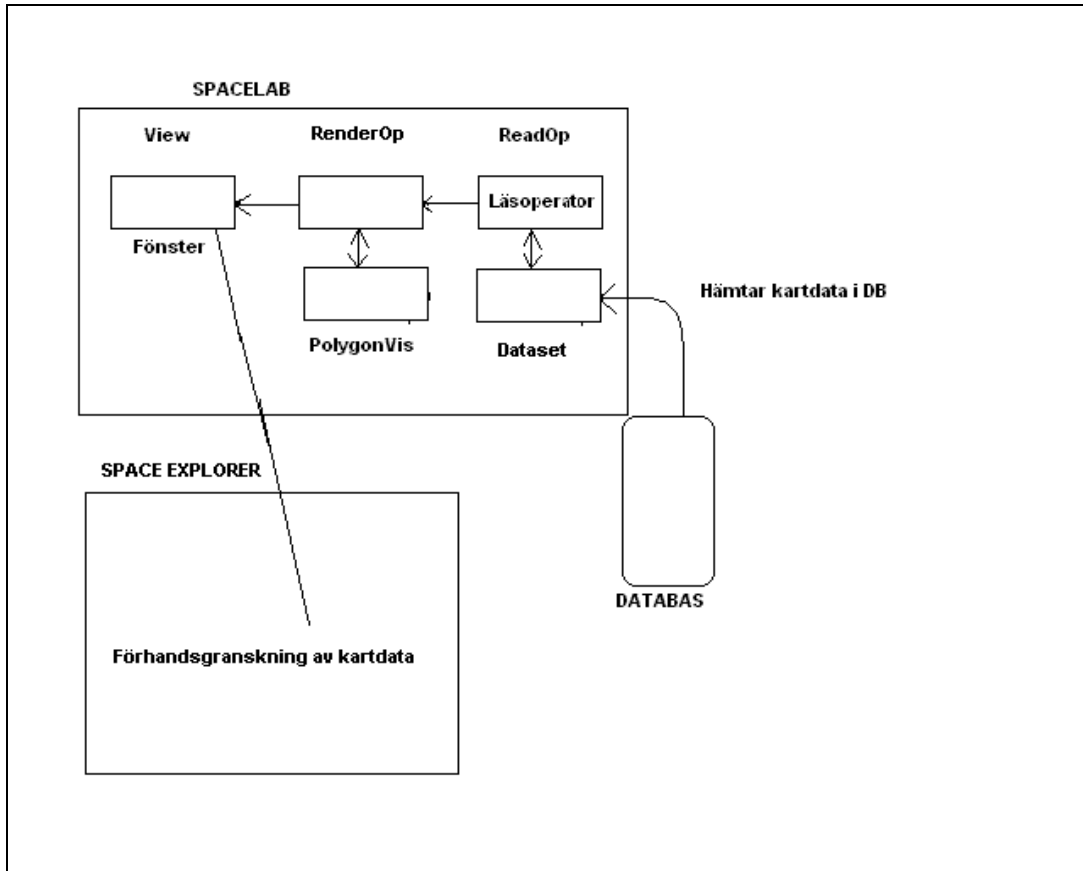
## **Avslut**

[När användaren är klar med enkäten]

Då får jag tacka för att du kom hit och ställde upp idag. [Ge användaren ersättningen]. Som ett bevis på att vi uppskattar att du hjälpt oss idag vill vi ge dig den här ersättningen. Har du några avslutande kommentarer om testet i stort?

Tack och hej då!

[Följ testdeltagaren till dörren]



En beskrivning av hur SpaceLab och dataflöden är upplagt.



## Appendix 4 - Medgivande till videoupptagning

### **Syfte**

Syftet med det här dokumentet är att nedteckna din tillåtelse att videofilma/logga dagens användbarhetstest. Anledningen att vi vill filma/logga sessionen är för att vid ett senare tillfälle djupare kunna analysera den informationen vi får idag. Det som sparas är i loggen är det som fångats upp av mikrofon och kamera, musklick, tangentryckningar samt vad som händer på skärmen. Loggningsfilen kommer enbart att användas internt inom Carmentas organisation för att utveckla den produkt testet berör, det vill säga att den inte kommer att användas för något annat syfte.

Om detta godkänns av dig, var vänlig att skriv under där så indikeras.

### **Medgivande**

Jag godkänner härmed att dagens användbarhetstest videofilmas för ovanstående syfte.

Namn: \_\_\_\_\_

Signatur: \_\_\_\_\_

Datum: \_\_\_\_\_

---

## Appendix 5 - Avslutningsenkät

Syftet med denna enkät är att hjälpa oss få en förståelse av de problem som en ny användare av SpaceLab upplever samt att få en uppfattning om användares första intryck av SpaceLab. Genom denna information kommer vi försöka förbättra SpaceLab och göra den lättare för nya användare att förstå och använda produkten. All information som framkommer är konfidentiell. Informationen kommer heller inte att användas till något annat syfte än att försöka förbättra användbarheten i SpaceLab.

### Om användaren:

#### 1. Ålder (välj en)

- Under 18
- 18 – 25
- 26 – 35
- 36 – 45
- 46 – 55
- 56 >

#### 2. Vad är din primära sysselsättning?

---

#### 3. Hur stora är dina programmeringskunskaper?

Obefintliga

Små

Stora

Programmerare

#### 4. Hur många poäng programmering har du läst? \_\_\_\_\_

Kommentar:

---

---

---

---

**Om SpaceLab:****5. Ange till vilken grad du håller med om följande påståenden.**

	Håller inte alls med	Håller inte med	Neutral	Håller med	Håller helt med	Vet inte
SpaceLab är lätt att använda	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Feedbacken är tillräcklig i SpaceLab	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SpaceLab är svårt att lära sig	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Den är enkelt att tappa bort sig	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Det krävs träning för att förstå SpaceLab	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Hjälpen är användbar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**6. Om du skulle få ändra tre saker i SpaceLab så att systemet blev lättare att komma igång med för nya användare, vad skulle det då vara?**

I. \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

II. \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

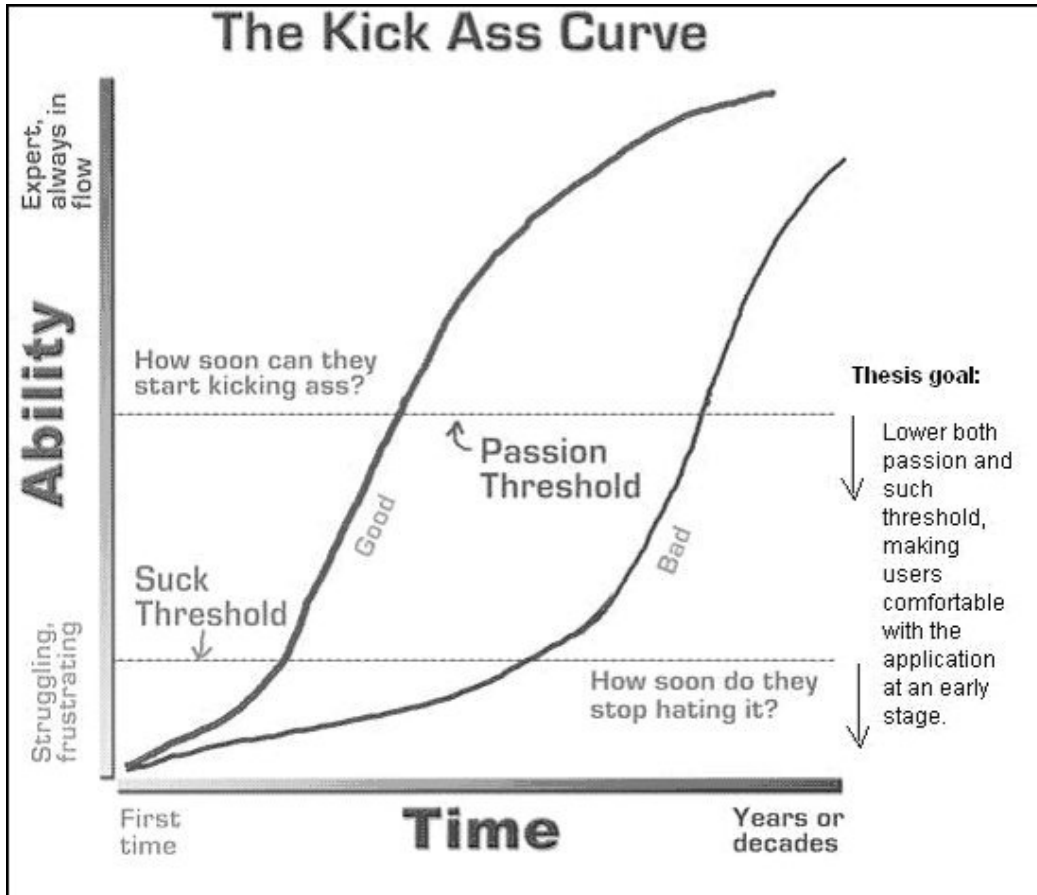
\_\_\_\_\_

III. \_\_\_\_\_



## Appendix 6 – Kick Ass Curve

The following chart explains (in somewhat questionable language) what the thesis goal has been, to lower the suck threshold for new users, allowing them to reach the passion threshold at an earlier stage than in the current process.



Original chart found at the blog: Creating passionate users, URL:

[http://headrush.typepad.com/creating\\_passionate\\_users/2005/10/getting\\_users\\_p.html](http://headrush.typepad.com/creating_passionate_users/2005/10/getting_users_p.html)