

THESIS / THÈSE

MASTER IN COMPUTER SCIENCE

Interactive visualization of usability data

Groetaers, David; Pagès, Nicolas

Award date:
2002

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.



FACULTÉS UNIVERSITAIRES NOTRE-DAME DE LA PAIX, NAMUR
INSTITUT D'INFORMATIQUE

ANNEE ACADEMIQUE 2001-2002

Interactive Visualization of Usability Data

David Groetaers and Nicolas Pagès

Thesis submitted in fulfilment of the requirements
for the degree of master in Computer Science

Supervisor : Prof. François Bodart

RUE GRANDGAGNAGE, 21 • B-5000 NAMUR (BELGIUM)

89077068 UBS

Abstract

Information visualisation is defined as “the use of computer-supported interactive visual representation of abstract data to amplify cognition” [Card et al., 1999]. This implies that information visualisation can be seen as a means for exploring data in a way that enhances our ability to understand relationships and identify patterns. Munzner maintains that the two key advantages of using computer-based systems for information visualisation are *interactivity* and *scalability* [Munzner, 2000].

Usability is “the effectiveness, efficiency and satisfaction with which a specified set of users can achieve a specified set of tasks in a particular environment” [ISO 9241, 1998]. Formal usability testing is an empirical usability evaluation method that requires the design of a formal usability experiment that is undertaken under controlled conditions in a usability laboratory [Nielsen, 1993]. It is aimed at measuring the different usability aspects of a system’s user interface and identifying specific problems with the interface. A usability laboratory is a formal observation environment that provides facilities for researchers to observe a user using a product, through logging the actions performed by the user, and videotaping the observation.

This dissertation is mainly concerned with the design and implementation of a fully functional visualisation system. This system, designed for expert users in usability testing, can be used to interactively visualise quantitative usability data resulting from tests performed in a usability laboratory.

The first step in the system’s usability engineering lifecycle was the analysis of competitive tools, that allow one to collect and visualise such data. The outcome is a list of usability problems we have tried to avoid in the creation of our system. Then, an investigation has been conducted to determine the semantics of the typical usability data that should be visualised. This investigation mainly resulted in the restriction of this data to certain aspects of quantitative data. Next, the functional and non-functional requirements that the system should fulfil have been expressed. A further goal was to design the interface and to determine the most suitable metaphor(s) to use to visualise this data with regard to the functional requirements. This point has led to the adoption of multiple views and the creation of several charts. These charts allow one to see the data from different perspectives, and thus to have its different aspects considered. The concept of interactivity was the central criteria when creating them, since it gives the user much freedom and control over the data to visualise. Next step consisted in determining the most appropriate implementation tool(s) to use. We have reached the conclusion that Java was an adequate language in the framework of this thesis, as well as for any kind of data visualisation. We finally performed a usability evaluation of the system we designed.

Keywords: Information visualisation, interactivity, multiple views, quantitative usability data, usability laboratory, usability testing.

Acknowledgments

There are a few people we would like to thank for their contribution to this project.

First, we are very grateful for the advice and support of our supervisor, François Bodart. His comments and insights have greatly improved and clarified this work. We also thank him for having offered us the opportunity to live a wonderful experience in such a beautiful country as South Africa.

Next, we would like to acknowledge everyone who provided support for this project and who facilitated our integration at the university of Port Elizabeth. Especially Professor Janet L. Wesson for initiating our research, for her guidance, advice and the time she gave us, and Lecturer Darelle van Greunen for her precious help and her attention towards us during our whole training at UPE.

We also would like to thank Professor Monique Noirhomme and Anne de Baenst for the literature they provided us.

Many friends in South Africa also contributed to the success of these five months. We would like to thank personally the Joubert family for their warm reception, their kindness and all the good time we had together. A special thanks also goes to Stephen Conradie for his generosity and the time he spent with us discovering Cape Town.

Lastly, we would like to thank our parents without whom nothing would have been possible. Thank you very much for your support since the first day.

Table of Contents

CHAPTER 1. INTRODUCTION	11
1.1 BACKGROUND.....	13
1.2 USABILITY LABORATORIES	14
The Observation Room and the Testing Room	14
Treatment of the participants	15
Videotaping.....	16
Cameraless Videotaping	17
1.3 PROBLEM STATEMENT	17
1.4 SITUATION OF CONCERN.....	18
1.5 STRUCTURE OF WRITE-UP.....	18
CHAPTER 2. LITERATURE REVIEW	21
2.1 INTRODUCTION	23
2.2 USABILITY	23
2.2.1 <i>What is Usability?</i>	24
Definitions	24
2.2.2 <i>Usability Engineering</i>	26
Description.....	26
Usability Engineering Lifecycle.....	27
2.2.3 <i>Usability Testing</i>	32
Usability inspection	32
Usability testing	32
Inspection methods	34
Testing methods.....	36
Usability laboratories related testing methods	38
Conclusion	41
2.2.4 <i>Usability Data Analysis</i>	41
Identify the most difficult tasks.....	41
Identify user errors and difficulties	41
Find the source of the error or difficulty	42
CHAPTER 3. AN ANALYSIS OF USABILITY DATA.....	43
3.1 INTRODUCTION	45
3.2 EXTANT SYSTEM ANALYSIS OF THE NOLDUS OBSERVER VIDEO PRO AND THE WEBTRENDS LOG ANALYSER	45
3.2.1 <i>Introduction</i>	45
WebTrends Overview	45
Noldus Overview	46
Why Noldus and WebTrends?	46
3.2.2 <i>Usability Analysis</i>	48
3.2.2.1 Introduction	48
3.2.2.2 Task Analysis	50
WebTrends.....	50
Noldus.....	53
3.2.2.3 Usability Analysis	61
WebTrends.....	61
Noldus.....	64
3.2.2.4 Visualisation evaluation	67
WebTrends.....	67
Noldus.....	71
3.2.3 <i>Data Analysis</i>	78
3.2.3.1 WebTrends	78
Extant data model (Class Diagram)	78
3.2.3.2 Noldus	79
Extant data model (OMT Notation)	79
3.2.4 <i>Conclusion</i>	81

3.2.4.1 WebTrends	81
3.2.4.2 Noldus	81
3.3. INVESTIGATION INTO THE SEMANTICS OF TYPICAL USABILITY DATA	82
3.3.1 Introduction	82
3.3.2 Qualitative and Quantitative Data	82
3.3.3 Common Characteristics	83
3.3.4 What is typical usability data	84
3.3.4.1 Different kinds of data	84
3.3.4.2 Conclusion	86
3.3.5 Typical quantitative usability data: case studies	86
3.3.5.1 WebTrends	86
3.3.5.2 Noldus	88
3.3.5.3 Typical Usability Data: General Schema	90
CHAPTER 4. AN ANALYSIS OF THE VISUALISATION REQUIREMENTS	93
4.1 INTRODUCTION	95
4.2 METHODOLOGY	95
4.3 LIMITATIONS OF THE PROJECT'S SCOPE	96
4.3.1 The users of the system	96
4.3.2 Type of data collected	96
4.3.3 Subjects and behaviors	96
4.3.4 Data sources	97
4.4 FUNCTIONAL REQUIREMENTS	97
4.4.1 The object of the visualisation	97
Effectiveness and efficiency	97
Other measurements	100
Conclusion	101
4.4.2 The visualisation techniques	103
4.4.3 Data access	103
4.4.4 Data handling	103
Overview Task	103
Zoom Task	103
Filter Task	104
Details-on-demand Task	104
Relate Task	104
History Task	105
Extract Task	105
4.4.5 Generalised task model	105
4.5 NON-FUNCTIONAL REQUIREMENTS	106
4.5.1 Usability	106
4.5.2 Reliability	107
4.5.3 Speed of performance	107
4.5.4 Design constraints	107
CHAPTER 5. DESIGN OF THE SYSTEM	109
5.1 INTRODUCTION	111
5.2 DATA REQUIREMENTS	111
5.2.1 Data collected	111
5.2.2 Data model	114
Class Diagram	114
Description	114
5.2.3 File format for the data	115
The Configuration File	115
The Observation File	119
5.3 THE DESIGN OF THE INTERFACE	123
5.3.1 Introduction	123
5.3.2 Multiple views	123
Definitions	123
When to use multiple views	123
Multiple views in our system	125

How to design multiple view systems.....	125
5.3.3 <i>An example of multiple view system: Spotfire DecisionSite</i>	127
Spotfire DecisionSite overview.....	127
Methodology for the analysis.....	127
Spotfire general design principles.....	128
Illustration of the functional requirements of our system with Spotfire.....	129
Results of the Spotfire analysis.....	135
5.3.4 <i>The interface</i>	136
Introduction.....	136
Space allocation in the main window.....	136
Selection and filtering panel.....	139
Visualisation panel.....	147
Menus.....	149
Toolbar.....	150
Interface overview.....	151
5.4 DESIGN OF THE VISUALISATION METAPHORS.....	152
5.4.1 <i>Introduction</i>	152
5.4.2 <i>General considerations</i>	152
5.4.3 <i>Design Criteria</i>	154
Title.....	154
Frame Dimensions.....	154
Scale.....	154
Grid lines.....	156
Messages.....	156
Colours.....	156
Three-dimensional features.....	158
5.4.4 <i>Identification and design of the visualisation metaphors</i>	159
Temporal sequence of behaviors.....	160
Time/Frequency per behavior.....	164
Sequential Analysis.....	165
Optimal Sequence of Behaviors.....	169
Statistics.....	172
Time Table.....	173
Video.....	173
5.4.5 <i>Design of the viewers</i>	173
2D graphs handling.....	174
3D graphs handling.....	174
Video Player.....	175
5.4.6 <i>Interaction between views</i>	177
5.4.7 <i>Conclusion</i>	178
CHAPTER 6. IMPLEMENTATION.....	179
6.1 INTRODUCTION.....	181
6.2 IMPLEMENTATION TOOL.....	181
6.3 IMPLEMENTATION.....	184
6.3.1 <i>Introduction</i>	184
6.3.2 <i>Class Diagram</i>	185
Legend.....	186
6.3.3 <i>Programming conventions</i>	187
Java Source Files.....	187
Indentation.....	188
Comments.....	188
Documentation Comments.....	188
Declarations.....	189
Statements.....	189
Naming Conventions.....	190
CHAPTER 7. EVALUATION OF THE SYSTEM.....	191
7.1 INTRODUCTION.....	193
7.2 HEURISTIC EVALUATION.....	195
7.2.1 <i>Introduction</i>	195
7.2.2 <i>The selection of observations</i>	195

7.2.3 <i>The data filtering</i>	197
7.2.4 <i>The data visualisation</i>	199
7.3 CONCLUSION.....	200
CHAPTER 8. CONCLUSION	201
8.1 ACHIEVEMENTS	203
8.2 FUTURE RESEARCH	205
APPENDICES	207
APPENDIX A: COMPLETE EXTANT TASK MODELS	209
<i>Complete Extant Task Model of the WebTrends Log Analyser.</i>	209
<i>Complete Extant Task Model of the Noldus Observer Video Pro.</i>	213
APPENDIX B: CLASS DIAGRAM NOTATION	219
<i>Class Diagram Notation</i>	219
<i>OMT notation</i>	222
APPENDIX C: BNF NOTATIONS FOR THE FILE FORMATS.....	225
<i>Configuration file</i>	225
<i>Observation file</i>	226
APPENDIX D: SPOTFIRE TEST FILE.....	229
BIBLIOGRAPHY	233

List of Figures

FIGURE 1: THE OBSERVER SIDE AT THE UPE'S USABILITY LABORATORY	15
FIGURE 2: MODULES OF THE OBSERVER.....	53
FIGURE 3: EVENT LOG.....	55
FIGURE 4: TIMERS.....	56
FIGURE 5: CHANNELS	56
FIGURE 6: BEHAVIOR CODES.....	56
FIGURE 7: TOOLBAR	57
FIGURE 8: THE OBSERVATION MODULE DURING A SCORING SESSION.....	57
FIGURE 9: DATA SELECTION.....	58
FIGURE 10: 2D COLUMN CHART.....	67
FIGURE 11: 3D COLUMN CHART.....	68
FIGURE 12: STACKED COLUMN CHART.....	68
FIGURE 13: AREA CHART	69
FIGURE 14: PIE CHART	70
FIGURE 15: EXAMPLE OF TIME EVENT TABLE	72
FIGURE 16: EXAMPLE OF TIME-EVENT PLOT	75
FIGURE 17: EXAMPLE OF ELEMENTARY STATISTICS.	76
FIGURE 18: EXAMPLE OF LAG SEQUENTIAL ANALYSIS.....	77
FIGURE 19: WEBTRENDS TYPICAL USABILITY DATA DIAGRAM.....	87
FIGURE 20: NOLDUS TYPICAL USABILITY DATA DIAGRAM	89
FIGURE 21: TYPICAL QUANTITATIVE USABILITY DATA DIAGRAM	91
FIGURE 22: DATA MODEL	114
FIGURE 23: SPOTFIRE GENERAL DESIGN PRINCIPLES	129
FIGURE 24: SPOTFIRE "QUERY DEVICES" INTERNAL FRAME.....	131
FIGURE 25: SPOTFIRE PIE CHART.....	131
FIGURE 26: SPOTFIRE HISTOGRAM	132
FIGURE 27: SPOTFIRE SCATTERPLOT	132
FIGURE 28: SPOTFIRE SCATTERPLOT 3D.....	133
FIGURE 29: SPOTFIRE TABLE	133
FIGURE 30: SPOTFIRE "DETAILS-ON-DEMAND" INTERNAL FRAME.....	134
FIGURE 31: SPACE ALLOCATION FOR THE MAIN WINDOW	139
FIGURE 32: SELECTION OF ALL OBSERVATIONS	140
FIGURE 33: THE "OVERVIEW" TAB	141
FIGURE 34: THE "GENDER" TAB	141
FIGURE 35: THE "AGE" TAB.....	142
FIGURE 36: THE "LANGUAGE" TAB.....	142
FIGURE 37: THE "EXPERTISE" TAB.....	143
FIGURE 38: SELECTION OF SOME OF THE OBSERVATIONS.....	143
FIGURE 39: THE COMPLETE DESCRIPTION OF AN OBSERVATION.....	145
FIGURE 40: THE SELECTION OF THE TIME PERIOD	146
FIGURE 41: THE SELECTION AND FILTERING PANEL: OVER VIEW	147
FIGURE 42: VISUALISATION PANEL WITH A UNIQUE VIEW.....	147
FIGURE 43: VISUALISATION PANEL WITH TWO VIEWS	148
FIGURE 44: VISUALISATION PANEL WITH THREE VIEWS	148
FIGURE 45: THE POP-UP MENU.....	150
FIGURE 46: THE TOOLBAR	151
FIGURE 47: THE INTERFACE: OVERVIEW	151
FIGURE 48: TEMPORAL SEQUENCE OF BEHAVIORS	162
FIGURE 49: DISPLAY OF THE SELECTED BEHAVIOR'S RELATED DATA.....	163
FIGURE 50: TIME/FREQUENCY PER BEHAVIOR	165

FIGURE 51: EXAMPLE OF MATRIX	166
FIGURE 52: SEQUENTIAL ANALYSIS.....	167
FIGURE 53: SEQUENTIAL ANALYSIS FOCUSING ON CERTAIN BEHAVIORS	168
FIGURE 54: OPTIMAL SEQUENCE OF BEHAVIORS	170
FIGURE 55: OPTIMAL SEQUENCE OF BEHAVIORS CHART WITH DIFFERENCES HIGHLIGHTED	171
FIGURE 56: DISPLAY OF THE SELECTED BEHAVIOR'S NAME	171
FIGURE 57: STATISTICS	172
FIGURE 58: TIME TABLE	173
FIGURE 59: 2D VIEWER	174
FIGURE 60: 3D VIEWER	175
FIGURE 61: VIDEO PLAYER.....	176
FIGURE 62: "OPEN MEDIA FILE" WINDOW	176
FIGURE 63: STRUCTURE OF THE APPLICATION	185

Chapter 1. Introduction

1.1 Background

Information visualisation is defined as “the use of computer-supported interactive visual representation of abstract data to amplify cognition” [Card et al., 1999]. This implies that information visualisation can be seen as a means for exploring data in a way that enhances our ability to understand relationships and identify patterns. Particularly challenging is the situation where large amounts of data are available. Tufte amongst others gives many examples of innovative and clever ways to visualise information but many of these are specific rather than generic [Tufte, 1990]. Munzner maintains that the two key advantages of using computer-based systems for information visualisation are *interactivity* and *scalability* [Munzner, 2000]. El ansari and Vanbrabant also stress that interactivity is a key principle of interactive information visualisation [El ansari and Vanbrabant, 2001].

Despite Tufte’s concern about the gratuitous use of 3D when no additional information is being conveyed, 3D does allow for a number of different effects to be represented [Tufte, 1990]. It allows for an extra dimension to be visualised with relative ease and enables the user to look at the information from various perspectives. In addition some chart types such as the InfoCube are inherently 3D [Darville and Van Espen, 2000]. People live in a 3D world and a 2D representation is to some extent artificial. The tools and techniques for displaying 3D information are now also available in the VRML and the Java3D packages.

According to Shneiderman, humans have remarkable perceptual abilities that are greatly under-utilised in current designs [Shneiderman, 1998]. Users can scan, recognise, and recall images rapidly and can detect subtle changes in size, colour, shape, movement or texture. User interfaces have been largely text oriented, so as visual approaches are explored, new opportunities for research are emerging. One key visual design guideline has emerged, however: *Overview first, zoom and filter, then details on demand*.

Formal usability testing is an empirical usability evaluation method that requires the design of a formal usability experiment that is undertaken under controlled conditions in a usability laboratory [Nielsen, 1993]. Evaluators give a user a specific task to perform. Evaluators observe the problem(s) the user has, videotape the user, and then analyse the observational logs and videotapes. UPE has recently installed the first usability laboratory (cf. section 1.2 below) of its kind in South Africa. The event logging software package installed at the UPE usability laboratory is the Observer Video Pro from Noldus Technology (March 2001). The Observer Video Pro allows for usability data to be captured and analysed live during an observation session or events to be coded from pre-recorded video. The usability data produced is mainly textual, however, and can be exported to an Excel spreadsheet for further analysis. A number of very simple 2D graphs can be produced, but the flexibility and information content of these is very poor. For example, a 2D time-event plot can be used to visualise the events that occurred during the usability session and a lag-sequential analysis displays a simple graph of the different event transitions. No facility exists to view overall usability data for a group of users and then be able to interactively zoom and filter this information.

1.2 Usability Laboratories

Typically, usability techniques involve observing a user as he or she uses an element of the product to determine the success of that element. Therefore many user tests take place in specially equipped usability laboratories. These laboratories are formal observation environments that provide facilities for researchers to observe a user using a product.

The Observation Room and the Testing Room

These testing environments can have multiple configurations, each of them having its own advantages and disadvantages, and corresponding to a different style or philosophy of testing. We will mainly focus on the classic laboratory setup, which is the one that is used in most laboratories for obvious reasons described below.

This setup consists of one room designated as the testing room and a second room designated as the observation room [Rubin, 1994]. A testing room (where the users perform tasks designed to test the device under investigation) is typically used to house the non-technical and technical equipment that the participants need to interact with a tested device, as well as the technical equipment that will be needed to capture their interactions. The only individual inside the testing room is the participant. All other test personnel, including the test monitor, observers, camera operator, etc., are stationed inside the observation room (see Figure 1). This room (where the researcher observes and records the users' interactions with the device under investigation) is generally used to house the non-technical and technical equipment that the researcher needs to record, observe and analyse the participants' interactions.

However, as an alternative, depending on the type of test, the test monitor could also be in the test room with the participant. Then, and as described in [Rubin, 1994], the test monitor has an excellent sense of what is going on with the participant, given the fact that he can see exactly what the participant is doing down to small mannerisms and subtle changes in body language. Furthermore, when the project is still at an early stage of its life cycle, and thus when much interaction is desired from the participant who is nearly considered as a partner helping to design the product, this position accentuates the sense of teamwork. But, and this is the main disadvantage of such a situation, the observer's behavior can affect the participant's behavior. For example, and even if the observer does not make any outward remark, the participant may pick up a sigh from him, and that might cause the participant to veer off from the direction he or she was following. And, of course, another problem is that this situation obviously limits the number of observers that can view the testing at the same time.

Therefore, and mainly because of the influence the observer might have on the participant, most usability laboratories are made of two distinguished rooms, and the participant stays alone in the testing room during the observation.

Those two rooms are ideally divided by a soundproof, one-way mirror, which allows the experimenters, other usability specialists and the developers to discuss user actions without disturbing the user.



Figure 1: The observer side at the UPE's usability laboratory

Treatment of the participants

As indicated in [Shneiderman, 1998], participants should always be treated with respect and should be informed that it is not they who are being tested; rather, it is the software and user interface that are under study. They should be told about what they will be doing and how long they will be expected to stay. Participation should always be voluntary, and informed consent should be obtained. The participants are often placed in awkward, stressful situations, where they have little control over events. The more they are put at ease, and the greater are the chances to have accurate results that applicable to real-world situations [Rubin, 1994]. For example, it is often better for the users to be shown the observation room and the different observers before the start of the test. It is indeed much less stressful for them to know who and what are behind the mirror, than having to imagine it [Nielsen 1993].

Another point underlined in [Rubin, 1994] is the benefit of being able to handle the participant's frustration during the test. In fact, it happens sometimes that the person becomes exceedingly frustrated during the course of a test. This should be interpreted as a critical point that could be advantageous. Indeed, since the frustration is usually related to a critical deficiency in the product, it is important for the test monitor not to yield to the temptation of asking the participant to give up and to go on to the next task for avoid confrontation, because in that case the risk would be to miss an important opportunity. Actually, "the participant's behavior at this point can be more revealing than at any other time", and can help the test team understand how the participant learns to use the product. Another advantage of encouraging the participant to keep trying is to show the designers and developers watching the test the difficulties that can be

experienced by participants using their product. This will probably do more to convince them to revise their product than any kind of discussion or negotiation.

Videotaping

A usability laboratory is usually equipped with several video cameras under remote control from the observation room. They can be used to show an overview of the test situation and to focus on the user's face, the keyboard, the manual and documentation, and the screen. A producer in the observation room then typically mixes the signal from these cameras to a single video stream that is recorded, and possibly time-stamped for later synchronisation with an observation log entered into a computer during the experiment. Indeed, reviewing videotapes is a tedious job, so careful logging and annotation during the test is vital to reduce the time spent finding critical incidents [Harrison, 1991]. Such synchronisation thus makes it possible to later find the video segment corresponding to a certain interesting user event without having to review the entire videotape [Nielsen, 1993]. This videotaping of the tasks performed is often valuable for later review and for showing designers or managers the problems that the users encountered [Lund, 1985].

We will notice that, according to [Nielsen, 1993], there is normally no need, for practical usability engineering purposes, to review a user test on videotape since one is mostly interested in finding major "usability catastrophes" anyway. These usability problems tend to be so glaring that they are obvious the first time they are observed and therefore do not require repeated perusal of a record of the test session. "This is especially true considering estimates that the time needed to analyse a videotape is between 3 and 10 times the duration of the original user test. In many cases, this extra time is better spent running more test subjects or testing more iterations of the design."

However, still according to [Nielsen, 1993], videotaping does have several uses in usability engineering. For example, a complete record of a series of user tests is a way to perform formal impact analysis of usability problems [Good et al., 1986]. Impact analysis involves first finding the usability problems and then going back to the videotapes to investigate exactly how many users had each usability problem and how much they were delayed by each problem. Since these estimates can only be made after one knows what usability problems to look for, an impact analysis requires a videotape or other detailed record of the test sessions. Alternatively, one can run more tests and count the known problems as they occur. Impact analysis can then be used to prioritise the fixing of the usability problems in a redesign such that the most effort is spent on those problems that are faced by many users and impact them severely.

Videotape also serves as an essential communications medium in many organisations where it may otherwise be difficult for human factors professionals to persuade developers and managers that a certain usability problem is in fact a problem. "Seeing a video of a user struggling with the problem often convinces these people" [Nielsen, 1993].

A final argument in favour of videotaping and equipment-extensive usability laboratories is the need to impress upper management and research funding agencies with the unique aspects of usability work. Some usability specialists feel that simpler techniques may be not sufficiently impressive to outsiders, whereas having an extensive laboratory will result in increased funding and respect due to its “advertising value” [Lindgaard, 1991].

Cameraless Videotaping

As described in [Nielsen, 1993], the main aspects of a test session can be captured on videotape without the use of cameras. Many computers provide a video output that either is directly compatible with video recording or can be made so by a scan converter. This video can be sent to the video recorder and thus allow the recording of the exact image the user sees on the monitor. This technique should result in better image quality than filming the monitor with a camera, but the video resolution will still be poorer than that of most computer monitors.

Cameraless videotaping has the obvious disadvantage of not including the user in the picture and not making it possible for a camera operator to zoom in on the interesting parts of the screen or the manual page being studied in vain by the user. Unless a high-definition television standard is used, one will also suffer a loss of resolution since current television standards use a poorer quality signal than that used by almost all computer monitors. These limitations make the resulting videotape less appealing and convincing in some cases.

More rarely, usability laboratories include other equipment to monitor users and study their detailed behavior. For example, an eyetracker can be used to collect data on what parts of the screens users look at [Benel et al., 1991].

1.3 Problem Statement

The goal of this project is to develop a visualisation tool that can be used to interactively visualise usability data resulting from tests performed in a usability laboratory. As a sub-goal of this project, an analysis of competitive tools that allow one to collect and visualise usability data has to be performed. As another sub-goal, an investigation has to be conducted to determine the semantics of typical usability data. The data should be accessed from a comma-delimited file or a standard database. This, among others, is specified in the functional and non-functional requirements that the system should fulfil. A further sub-goal is to design the interface and to determine the most suitable metaphor(s) to use to visualise this data. A final sub-goal is to determine the most appropriate implementation tool(s) to use to implement the visualisation tool.

1.4 Situation of Concern

Usability data visualisation is a topic that has been little addressed until now. Many authors barely evoke it in their works, and it is thus quite difficult to find something about it in the literature. Most often, the visualisation field is discussed, but from a general perspective, i.e. for any kind of information. Different techniques used to analyse usability data are available, but they mainly consist of theoretic procedures. Different usability data collection methods exist, and most of them suppose the existence of predefined goals allowing accurate assessment of the system, but no visual ways of doing that are proposed.

1.5 Structure of Write-up

In Chapter 2, we present a comprehensive literature review of the usability field, which is composed of different sections. The first one is intended to give general overview of the usability concept, through exposing several definitions that have been given by different authors. The second part more particularly focuses on usability engineering, and contains a summary of the usability engineering lifecycle model that can be found in [Nielsen, 1993].

Chapter 3 is divided in two main parts. The first is called “Extant System Analysis of Noldus Observer Video Pro and WebTrends Log Analyser”. Its purpose is to bring into light the main qualities and defaults of these two systems through a heuristic evaluation¹. They are both programs allowing collection and analysis of usability data, and they can therefore be interesting sources of information with regard to the problem statement. The second part, that deals with the first sub-goal of the thesis, is an investigation into the semantics of typical usability data. Many kinds of data can result from usability evaluation. These can be different at the level of their nature as well as the types of conclusions they allow experimenters to draw. We try to determine, for all those kinds of data, which are their common characteristics, and among them, which are the more representative and speaking ones.

Chapter 4 has the intent of determining the main requirements of the visualisation tool we want to develop. Basically, these requirements are divided into two main groups: the functional requirements and the non-functional requirements. As described in [Rubin, 1994], the functional requirements describe the product’s intended functionality as well as the tasks the end user will perform. The other specifications of the system are covered in the non-functional requirements section. These include criteria such as reliability, speed of performance, etc.

¹ See Chapters 2 and 3 for more details about the heuristic evaluation.

Chapter 5 is about the design of the system. This large chapter describes several aspects of the design, among which one can mention the input files' formats, the design of the interface, and a discussion explaining the choices we made to address the second sub-goal of the thesis: The determination of the most suitable metaphor(s) to use to visualise the data defined in Chapter 4.

Chapter 6 is more technical since it discusses the implementation of the application. The choice of the implementation tool is explained, as well as the programming conventions followed during that implementation. The main components of the application's internal structure are exposed too.

Chapter 7 presents a heuristic evaluation of the system, which is intended to assess its usability. This evaluation is based on the tasks defined in chapter 4.

Chapter 8 regroups all the aspects addressed all along this thesis, and suggests proposals for future research.

Chapter 2. Literature Review

2.1 Introduction

The purpose of this chapter is to describe different aspects of the field of usability, such as usability testing and usability engineering. We first present a description of usability, through different definitions available in the literature, then we go through a complete usability engineering lifecycle, before presenting the main usability evaluation methods. To end, we consider the process to follow when analysing usability data. For a literature review in the field of visualisation, see Chapter 5 (section 5.4) in which we present some guidelines related to graphic design.

2.2 Usability

In the past, when computers were only used by a small number of people who mostly performed very specialised tasks, it made sense to require a high degree of learning and expertise from the users. But now the revolution in personal computers and falling hardware prices have made computers available to ever growing groups of users, and these users are using computers for a large variety of tasks. Therefore user interfaces have become a much more important part of computers than they used to be [Nielsen, 1993].

Today, as a consequence of this phenomenon, one of the biggest problems that designers of a human computer system face when developing a system is making sure that the finished product is what the user really wants and needs. It is far from easy to ensure that a product is suitable for the purpose for which it was designed given the complicated specifications many computer systems are endeavouring to fulfil. In order to answer this growing need for usability, which follows logically the growing number of people using computers and the systems developed for them, different techniques and theories have been developed around the field of usability, with the objective to make software usable by most people.

Usability engineering is the name commonly given to the systematic approach consisting in trying to make software easier to use for the individuals who actually use it in their work. It is an evolving science that determines best practices and continually tests and refines its techniques. But usability engineering not only represents the techniques, processes, methods and procedures for designing usable products and systems, but, just as important, the philosophy that places the user at the centre of the design [Rubin, 1994].

As described in [van Greunen, 2002], the difference between usability engineering and usability testing is that usability engineering is a methodological approach to producing user interface, and a way to deliver a product that works, when usability testing is only a part of the process of usability engineering, involving real users performing real tasks.

Usability testing is a methodology for measuring the different usability aspects of a system's user interface and identifying specific problems with the interface. It is an increasingly important part of the overall user interface iterative design process, which consists of designing, prototyping and evaluation, and it is itself a process that entails many activities: specifying evaluation goals, identifying target users, selecting usability metrics, selecting an evaluation method and tasks, designing experiments, collecting usability data, and analysing and interpreting data. [Shneiderman, 1998] says about the emergence of usability testing and laboratories since the early 1980s that it "is an indicator of the profound shift in attention in user needs. Traditional managers resisted at first, saying that usability testing seemed like a nice idea, but that time pressures or limited resources prevented them from trying it. As experience grew and successful projects gave credit to the testing process, demand swelled and design teams began to compete for the scarce resource of the usability-laboratory staff. Managers came to realise that having a usability test on the schedule was a powerful incentive to complete a design phase. The usability-test report provided supportive confirmation of progress and specific recommendations for changes. Designers sought the bright light of evaluative feedback to guide their work, and managers saw fewer disaster projects approached delivery dates. The remarkable surprise was that usability testing not only sped up many projects, but also produced dramatic cost savings." [van Greunen, 2002] indicates that usability potential benefits include decreased late design changes, decreased user training, increased user productivity, decreased user errors, and decreased need for user support.

2.2.1 What is Usability?

First, we can make a distinction between utility and usability: utility is the question of whether the functionality of the system in principle can do what is needed, and usability is the question of how well users can use that functionality [Grudin, 1992]. In other words, for the user, good usability of a computer program means that he or she can work efficiently and with ease. Usability thus applies to all aspects of a system with which a human might interact, including installation and maintenance procedures [Nielsen, 1993].

Definitions

As said in [Bevan et al., 1991], there are still different approaches to making a product usable, and yet no accepted definition of the term usable. The definitions which have been used derive from a number of views of what usability is. Three of the views relate to how usability should be measured:

- The product-oriented view, that usability can be measured in terms of the ergonomic attributes of the product.
- The user-oriented view, that usability can be measured in terms of the mental effort and attitude of the user.

- The user performance view, that usability can be measured by examining how the user interacts with the product, with particular emphasis on either
 - ease-of-use: how easy the product is to use, or
 - acceptability: whether the product will be used in the real world.

These views are complemented by the contextually oriented view, that usability of a product is a function of the particular user or class of users being studied, the task they perform, and environment in which they work.

- For example, the definition given in the ISO standard for software qualities [ISO 1991b] is product and user-oriented: “a set of attributes of software which bear on the effort needed for use and on the individual assessment of such use...”
- The proposed ISO ergonomics definition [ISO 9241, 1998; Brooke et al., 1990] is usage, user and contextually oriented: “the effectiveness, efficiency and satisfaction with which a specified set of users can achieve a specified set of tasks in a particular environment”. This definition suggests that a usable system enables its users to perform their computer-related tasks accurately and completely (effectiveness) and doing so with ease of learning, use and recollection (efficiency), while fulfilling their task-related requirements (satisfaction). We will come back later on these three very important concepts (cf. Ch.3).
- The definition given in [Eason, 1988] is ease-of-use oriented: “the degree to which users are able to use the system with the skills, knowledge, stereotypes and experience they can bring to bear”.
- The position taken by the ESPRIT MUSiC² project is that a complete definition of usability must encompass all these views. Usability is a function of the ease of use (including learnability when relevant) and the acceptability of the product and will determine the actual usage by a particular user for a particular task in a particular context. The current MUSiC definition of usability is: the ease of use and acceptability of a system or product for a particular class of users carrying out specific tasks in a specific environment; where ‘ease of use’ affects user performance and satisfaction, and ‘acceptability’ affects whether or not the product is used. Ease of use determines whether a product can be used, and acceptability whether it will be used, and how it will be used. Ease of use in a particular context is determined by the product attributes, and is measured by user performance and satisfaction. The context consists of the user, task and

² The *MUSiC performance measurement method* [Macleod et al., 1994] was developed by the European MUSiC (**M**etrics for **U**sability **S**tandards in **C**omputing) project to provide valid and reliable means of specifying and measuring usability. The method gives feedback on how to improve the usability of the design. MUSiC also includes tools and techniques for measuring user performance and satisfaction. Products supporting the method are available, e.g. DRUM [Macleod & Rengger, 1993], a video analysis software.

physical and social environment. Acceptability is defined as a combination of social and practical acceptability [van Greunen, 2002]. Practical acceptability consists of several factors: cost, support, reliability, compatibility, and usefulness (measure of whether the system can achieve a desired goal).

- It is said in [Nielsen, 1993] that it is important to realise that usability is not a single, one-dimensional property of a user interface. Usability has multiple components and is traditionally associated with these five usability attributes:
 - Learnability: The system should be easy to learn so that the user can rapidly start getting some work done with the system.
 - Efficiency: The system should be efficient to use, so that once the user has learned the system, a high level of productivity is possible.
 - Memorability: The system should be easy to remember, so that the casual user is able to return to the system after some period of not having used it, without having to learn everything all over again.
 - Errors: The system should have a low error rate, so that users make few errors during the use of the system, and so that if they do make errors they can easily recover from them. Further, catastrophic errors must not occur.
 - Satisfaction: The system should be pleasant to use, so that users are subjectively satisfied when using it; they like it.

2.2.2 Usability Engineering

Description

“Have you ever seen one of the people who will be users of your current project? Have you talked to such a user? Have you visited the users’ work environment and observed what their tasks are, how they approach these tasks, and what pragmatic circumstances they have to cope with?” [Nielsen, 1993]. According to Nielsen, such simple user-centred activities form the basis of usability engineering.

Usability engineering is a composed set of activities that ideally take place throughout the lifecycle of the product, with significant activities happening at the early stages before the user interface has even been designed. The need to have multiple usability engineering stages supplement each other was recognised early in the field, though not always followed on development projects [Gould and Lewis, 1985].

We can add that usability engineering is a process, grounded in classical engineering, which amounts to specifying, quantitatively and in advance, what characteristics and in what amounts the final product to be engineered is to have [Good et al., 1986]. Without measurable usability specifications, there is no way to determine the usability needs of a product, or to measure whether or not the finished product fulfils those needs. If we cannot measure usability, then we cannot have usability engineering. As said in [van Greunen, 2002], usability engineering thus refers to the entire process of producing usable software products from requirements gathering to installation.

Usability Engineering Lifecycle

Usability cannot be seen in isolation from the broader corporate product development context where one-shot projects are fairly rare. Indeed, usability applies to the development of entire product families and extended project where products are released in several versions over time. In fact, this broader context only strengthens the arguments for allocating substantial usability engineering resources as early as possible, since design decisions made for any given product have ripple effects due to the need for subsequent products and versions to be backward compatible [Nielsen, 1993]. Consequently, some usability engineering specialists [Grudin et al., 1987] believe that “human factors involvement with a particular product may ultimately have its greatest impact on future product releases.” Planning for future versions is also a prime reason to follow up the release of a product with field studies of its actual use.

Here are the different stages of the usability engineering lifecycle model [Nielsen, 1993]:

1. Know the user

The first step in the usability process is to study the intended users and use of the product. And it is important to keep in mind that all users should be considered, which include installers, maintainers, system administrators, and other support staff. It thus means that the concept of user should include everybody whose work is affected by the product in some way.

- a. Individual user characteristics

It is necessary to know the class of people who will be using the system. This supposes to know the users' work experience, educational level, age, previous computer experience, etc. All that information may help anticipate the users' learning difficulties, and to better set appropriate limits for the complexity of the user interface.

- b. The user's current and desired tasks

A task analysis is essential as early input to system design. The users' overall goals should be studied as well as how they currently approach the task, their information needs, and how they deal with exceptional circumstances or emergencies. A typical outcome of a task analysis is a list of all the things the users want to accomplish with system, all the information they will need to achieve these goals, the steps that need to be performed and the interdependencies between these steps. Often, a task analysis can be decomposed in a hierarchical fashion [Grief, 1991], starting with the larger tasks and goals of the organisation and breaking each of them down into smaller subtasks, that can again be further subdivided.

c. Functional analysis

A new computer system should not be designed only to propagate sub-optimal ways of doing things that may have been instituted only because of limitations in previous technologies. Therefore, it is necessary not to just analyse the way users currently do the task, but also the underlying functional reason for the task: what is it that really needs to be done, and what are merely surface procedures which can and probably should be changed [Schmidt, 1988].

d. The evolution of the user and the job

Using the system will change the users, and as they change they will use the system in new ways. It is impossible to forecast these changes completely as users will always discover new uses for computer systems after some period of use, but a flexible design will stand a better chance of supporting these new uses. A typical change is that users become experts after some time and want interaction shortcuts.

2. Competitive analysis

As said further, prototyping is an important phase of the usability process, and already existing products are often the best prototypes to get from a product [Byrne, 1989]. It is thus desirable to analyse existing products heuristically according to usability guidelines and to perform empirical user tests with these products. A competing product is already fully implemented and can therefore be tested very easily [Bachman, 1989]. This means that users can perform real tasks on the competing system, and therefore make more realistic tests than on prototypes. Furthermore, if several competing products are available for analysis, it is then possible to perform a comparative analysis which can provide ideas for the design and a list of guidelines that seem to work or that should be avoided.

3. Setting usability goals

As said above, usability comprises several elements that can sometimes conflict. Now, all those aspects cannot be given equal weight in a given design project, and it is thus necessary to make choices and priorities based on the analysis of the users and their tasks. For each usability attribute of interest, several different levels of performance can be specified as part of a goal-setting process [Whiteside et al., 1988]. One would at least specify the minimum level which would be acceptable for release of the product, but a more detailed goal specification can also include the planned level one is aiming for as well as the current level of performance. Usability goals are reasonably easy to set for new versions of existing systems: the minimum acceptable usability would normally be equal to the current usability level, and the target usability could be derived as an improvement that was sufficiently large to induce users to change systems.

a. Financial impact analysis

During the same period usability goals are being specified, it is interesting to make an analysis of the financial impact of the usability of the system. Such an analysis involves estimating the number of users who will be using the system, their loaded salaries or other costs, and the approximate time they will be using the system. The cost of the users' time is not just their salary but also other costs, such as various pensions and benefits, employment taxes or fees charged by the government, and general overhead costs like the rent of office space.

4. Parallel design

It is often a good idea to start the design with a parallel design process, in which several different designers work out preliminary designs [Nielsen et al, 1993-94]. The goal is to explore different design alternatives before settling on a single approach that can be developed in further detail and subjected to more detailed usability activities.

5. Participatory design

Even though the advice to "know the user" may have been followed since the beginning of the design phase, one still cannot know the user sufficiently well to answer all issues that come up in doing the design [Kensing and Munk-Madsen, 1993]. Instead of guessing, designers should have access to a pool of representative users after the start of the design phase. Such users often raise questions that the development team has not thought of asking. Therefore, users should be involved in the design process through regular meetings between designers and users.

6. Coordinated design of the total interface

Consistency is one of the most important usability characteristics. It should apply across the different media that form the user interface, including not only the applications screens but also the documentation, the online help system, etc [Perlman, 1989]. Consistency is not just measured at a single point of time but should apply over successive releases of a product so that new releases are consistent with their predecessors.

7. Apply guidelines and heuristic analysis

Guidelines list well-known principles for user interface design which should be followed in the development project. In any given project, several different levels of guidelines should be used: general guidelines, applicable to any user interface, category-specific guidelines for the kind of system being developed (e.g. guidelines describing how to display multi-views, guidelines that can be applied to the design of charts, etc.), and product-specific guidelines for the individual product.

8. Prototyping

Early evaluation can be based on prototypes of the final systems that can be developed much faster and much more cheaply, and which can thus be changed many times until a better understanding of the user interface design has been achieved. The entire idea behind prototyping is to save on the time and cost to develop something that can be tested with real users. These savings can only be achieved by somehow reducing the prototype compared with the full system. This can be done through different techniques: placing less emphasis on the efficiency of the implementation, accepting less reliable or poorer quality code, using simplified algorithms that cannot handle all the special cases, using fake data and other content, etc.

9. Empirical testing

[Whitefield et al., 1991] provide a classification of evaluation methods on two dimensions: whether or not real users are involved and whether or not interface has actually been implemented. It is normal to expect better results from testing real users and real systems, but doing so may not always be feasible. The prototyping methods described above provide a means of performing evaluations early enough to influence a project while it can still change direction, and the heuristic evaluation method described in the next section (Usability Testing) allows testers to assess usability without the expense of a user test. From whatever evaluation methods are used, a major result will be a list of usability problems in the interfaces as well as hints for features to support successful user strategies. It is normally not feasible to solve all problems, so it will probably be necessary to prioritise them. Those priorities are best based on experimental data about the impact of the problems on user performance, but sometimes one has to rely on subjective judgements only.

10. Iterative design

Based on usability problems and opportunities disclosed by the empirical testing, it is possible to produce a new version of the interface. Some testing methods such as thinking aloud, log files of user interaction sequences, knowledge of usability guidelines, can suggest specific changes to the interface. Familiarity with the design options, insight gained from watching users, creativity, and luck are needed at this point. Indeed, sometimes some of the changes made to solve certain usability problems may fail to achieve this objective, and may even introduce new usability problems [Bailey, 1993]. This is a good reason for combining iterative design and evaluation. In fact, it is quite common for a redesign to focus on improving one of the usability parameters, only to find that some of the changes have impacted other usability parameters. The time and expense needed to fix a particular problem is also a factor in determining priorities. Furthermore, it is likely that additional usability problems appear in repeated tests after the most blatant problems have been corrected. Therefore the initial designs, since they will be changed anyway, should not be tested comprehensively, but changed and re-tested as soon as a usability

problem has been detected and understood, so that those remaining problems masked by the initial glaring ones can be found.

a. Capture design rationale

The rationale for the various user interface design decisions can be made explicit and recorded for later reference [Moran and Carroll, 1994]. Having access to an audit trail through the design rationale is important during iterative development and during development of any future releases of the product. Since changes to the interface will often have to be made, it is helpful to know the reasons underlying the original design so that important usability principles are not sacrificed to attain a minor objective. Furthermore, the design rationale can help in maintaining user interface consistency across successive product versions.

11. Collect feedback from field use

The main objective of usability work after the release of a product is to gather usability data for the next version and for new, future products: in the same way existing and competing products were the best prototypes for the product in the initial competitive analysis phase, a newly released product can be viewed as a prototype of future products. Studies of the use of the product in the field assess how real users use the interface for naturally occurring tasks in their real-world working environment and can therefore provide much insight that would not be easily available from laboratory studies. Field feedback can be gathered from active seeking of the users as well as from more passive techniques such as analysing user complaints or modification requests.

This model emphasises that one should not rush straight into design. “The least expensive way for usability activities to influence a product is to do as much as possible before design has started, since it will then not be necessary to change the design to comply with the usability recommendations” [Nielsen, 1993]. Also, usability work done before the system is designed may make it possible to avoid developing unnecessary features.

2.2.3 Usability Testing

We will first establish a triple distinction in the level of the different techniques available in order to achieve a usability test. The first distinction will be between usability inspection and usability testing:

Usability inspection

- Usability inspection is the generic name for a set of methods based on having evaluators inspect or examine usability-related aspects of a user interface. Usability inspectors can be usability specialists, but they can also be software development consultants with special expertise (e.g., knowledge of a particular interface style for graphical user interfaces), end users with content or task knowledge, or other types of professionals. The different inspection methods have slightly different goals, but normally, usability inspection is intended as a way of evaluating user interface designs. In usability inspection, the evaluation of the user interface is based on the considered judgement of the inspector(s). The individual inspection methods vary as to how this judgement is derived and on what evaluative criteria inspectors are expected to base their judgements. In general, the defining characteristic of usability inspection is the reliance on judgement as a source of evaluative feedback on specific elements of user interface [Nielsen and Mack, 1994].
- Given that usability inspections are part of a larger development process, it is also important to consider under what conditions inspections are appropriate and most effectively applied. Normally, inspection methods are not suited for use in the early stages of the usability engineering lifecycle where no user interface has been designed or implemented. Similarly, inspection methods are poorly suited for usability engineering very late in the cycle, when the system has been released to customers and the goal is to have them actually use it and get ideas for future versions. In general, inspection methods require the existence of an interface design, not necessarily implemented. Indeed, many of these methods do not require the inspector to actually use the system. This means that inspections can be done relatively early in the design stage, compared to usability testing.

Usability testing

- Usability testing, which involves real users, is the most fundamental usability method and is in some sense irreplaceable, since it provides direct information about how people use computers and what their exact problems are with the interface being tested. There are several methodological pitfalls in usability testing, as in all kinds of testing, and one needs to pay attention to the issues of reliability and validity. Reliability is the question of whether one would get the same result if the test were to be repeated, and validity is the question of whether the result actually reflects the usability issues one wants to test. [Nielsen, 1993].

- Before any testing is conducted, one should clarify the purpose of the test since it will have significant impact on the on the kind of testing to be done. A major distinction is whether the test is intended as a formative or summative evaluation of the user interface. Formative evaluation is done in order to help improve the interface as part of an iterative design process. The main goals of formative evaluation are thus to learn which detailed aspects of the interface are good and bad, and how the design can be improved. A typical method is the thinking-aloud test. In contrast, summative evaluation aims at assessing the overall quality of an interface, for example for use in deciding between two possible designs or as part of a competitive analysis to learn how good the competition really is [Nielsen, 1993].
- There are some limitations to testing. Indeed, testing does not guarantee success or even prove that a product will be usable. This is due to different reasons, among which we can give the following ones: testing is always an artificial situation; participants are rarely fully representative of the target population; test results do not formally prove that a product is usable, but they constitute a measure of the probability it does. However, in spite of these limitations, usability testing, when conducted with care and precision, for the appropriate reasons, at the appropriate time in the product development lifecycle, and as part of an overall user-centred design approach, is an almost infallible indicator of potential problems and the means to resolve them. Therefore, in almost every case, “it is better to test than not to test” [Rubin, 1994].

The second distinction is to be made at the level of the usability test’s environment [Thomson]:

- Formal laboratories: they are structured environments that tend to seclude the user into situations that do not necessarily reflect the real circumstances in which the tested object will be used.
- Informal laboratories environments: such laboratories contain elements of both formal laboratories and real world environments. The laboratory thus features elements of the “real world” situation in which the object is normally used.
- Real world environments: they are the real physical environments in which the object is used.

The third distinction implies the kind of data that results from the analysis: either qualitative data or quantitative data. The difference between those two kinds of data is discussed more deeply in the next chapter.

We have chosen to present the different usability evaluation methods in groups corresponding to the first distinction, i.e. to make a distinction between the inspection methods and the testing methods. However, since many usability tests take place in usability laboratories, we have decided to present those in a separate a sub-section called “Usability laboratories related testing methods”.

Inspection methods

Heuristic evaluation

It is the most fundamental method and it involves having (a small set of) usability specialists judge whether each dialog element conforms to established usability principles. These principles are normally referred to as the heuristics and give the method its name [Nielsen and Mack, 1994]. In general, this evaluation is difficult for a single individual to do because one person will not be able to find all the usability problems in an interface. However, experience has shown that different people generally find different usability problems, and it is thus possible to increase the effectiveness of the method significantly by involving multiple evaluators. A heuristic evaluation is performed by having each individual evaluator inspect the interface alone. Only after all observations have been completed are the evaluators allowed to communicate and have their findings aggregated. This procedure is important in order to ensure independent and unbiased evaluations from each evaluator [Nielsen and Mack, 1994]. We will add that the output from using the heuristic evaluation method is a list of usability problems in the interface, annotated with references to those usability principles that were violated by the design. Heuristic evaluation thus does not provide a systematic way to generate fixes to the usability problems. However, because each problem is explained to reference established usability principles, it will often be fairly easy to generate a revised design according to the guidelines provided by the dialogue principle that was violated [Nielsen, 1993].

Guideline review

A guideline review is an inspection method where an interface is checked for conformance with a comprehensive list of usability guidelines. However, since guideline documents generally contain on the order of thousands of guidelines, a guideline review requires a high level of expertise and is fairly rare in practise [Nielsen and Mack, 1994].

Standards inspection

A standards inspection has an expert on some interface standard (e.g. the OSF/Motif standard) inspect the interface for compliance. Thus, standards are aimed at increasing the degree to which a given interface is in the range of other systems on the market that follow the same standards [Nielsen and Mack, 1994].

Pluralistic walkthrough

It is a meeting where representative users, product developers, and human factors professionals step through a scenario, discussing usability issues associated with dialogue elements involved in the scenario steps. The scenario entails one linear path through a series of user interface panels. During the walkthrough, the participants confront the panels just as they would during the successful conduct of the specified task in any other context. All participants are asked to assume the role of the user, whatever user population is being tested at the time. It is important to underline that the discussion begins only after the task has been “completed” and all participants have written down the choices they would have made for each different panel [Rubin, 1994].

Cognitive walkthrough

Cognitive walkthrough involves researchers learning how the users understand and perceive the object tested. The experts simulate users walking through the interface in the context of one or more specific user tasks. They follow a detailed procedure to simulate a user’s problem-solving process at each step in the human-computer dialogue, checking to see if the simulated user’s goals and memory for actions can be assumed to lead to the next correct action [Nielsen and Mack, 1994]. For each action, the analysts wonder what the user would be trying to do at that point and what actions the interface makes available. If the interface design is a good one, then the user’s intentions should cause that person to select the appropriate action. Following the action, the interface should present clear feedback indicating that progress is being made toward completing the task [Rubin, 1994]. High-frequency tasks are a starting point, but rare critical tasks, such as error recovery, should also be walked through [Shneiderman, 1998]. The cognitive walkthrough focuses on detecting and evaluating the design’s defects, particularly by exploration. This focus is motivated by the observation that many users prefer to learn software by exploration [Carroll and Rosson, 1987].

Consistency inspection

The consistency inspections can be considered on a double level: first, they are aimed at producing the maximum degree of consistency among all components of a system within the time constraints of the development schedule. And second, they may have designers representing multiple projects inspect an interface to see whether it does things in a way consistent with their own design. Consistency inspections are aimed at evaluating consistency for the different parts of the same product, and across the family of products that has been evaluated by an inspection team [Nielsen and Mack, 1994].

Feature inspection

This inspection focuses on the function delivered in a software system. It can determine, for example, whether the function as designed meets the needs of the intended end users. A feature inspection can involve not only evaluation of the utility of a function, but can also involve the design of that function.

Testing methods

Observation

Simply visiting the users to observe them work is an extremely important usability method with applications both for the task analysis and for information about the true field usability of installed systems [Diaper, 1989]. Observation is really the simplest of all usability methods since it involves visiting one or more users and then doing as little as possible in order not to interfere with their work. Of course, the observer may take notes (unobtrusively), and it may even be possible to use videotaping in some environments, though most computer customers do not like to have outsiders come in and videotape their business.

The observer's goal when conducting an observation is to become virtually invisible to the users so that they will perform their work and use the system in the same way they normally do. Every now and then, it may become necessary to interrupt a user to ask for an explanation of some activity that is impossible for the observer to understand, but such questions to the users should be kept to the minimum.

Since the observer will normally be an expert in using the tested system, the users will probably have many questions to ask to the observer, and they may request help in getting the system to do certain tasks. During the beginning of the visit, the observer should decline any such requests for assistance, giving the explanation that he or she is there to observe how the users work when they do not have a systems expert around.

One advantage of observing users doing their own tasks is that one often finds that they use the software in unexpected ways that one would not have sought to test in a planned laboratory experiment [Nielsen, 1993].

Focus groups

Focus groups are a somewhat informal technique that can be used to address user needs and feelings both before the interface has been designed and after it has been in use for some time, to discover users' attitudes and cognitive associations to new ideas and designs, or to assess the user's object requirements. In a focus group, usually, about six to nine users are brought together to discuss new concepts and identify issues over a period of about two hours. Each group is run by a moderator who is responsible for maintaining the focus of the group on whatever issues of interest. From the users' perspective, a focus group session should feel free-flowing and relatively unstructured, but in reality, the moderator has to follow a pre-planned script for what issues to bring up.

Focus groups often bring out users' spontaneous reactions and ideas through the interaction between the participants and have the major advantage of allowing observation of some group dynamics and organisational issues [Nielsen, 1993].

Questionnaires and interviews

Many aspects of usability can best be studied by simply asking the users. This is especially true for issues relating to the user's subjective satisfaction and possible anxieties, which are hard to measure objectively. Questionnaires and interviews are also useful methods for studying how users use systems and what features they particularly like or dislike. It is important to notice that user statements should not always be taken at face value. For example, in a classic study [Root and Draper, 1983], a comparison of the results from questionnaires administered before and after the introduction of some new features to the system gave a significant result. Indeed, the correlation between users' predictions of whether they would like the new features and their ratings of the features after having tried them was only 0.28, indicating that one should not always interpret the results literally when asking users about interface elements they have not tried.

Questionnaires and interviews are similar methods since both involve asking users a set of questions and recording their answers. The difference is that questionnaires are printed on paper or presented interactively on a computer and can be administered without the need of having any other people present beside the user answering the questions. In contrast, interviews involve having an interviewer read the questions to the respondent, and the answers are recorded by the interviewer instead of being filled by the respondent. Interviews thus require more usability staff time, but they do have the advantage of being more flexible, since the interviewer can ask more difficult questions in more depth and can rephrase a question if the respondent's answer indicates that the question was misunderstood [Nielsen, 1993].

User feedback

For installed systems, the users themselves can form a major source of usability information if one is willing to listen to their feedback. User feedback has several advantages:

- It is initiated by the users, so it shows their immediate and pressing concerns
- It is an ongoing process, so feedback will be received without any special efforts to collect it.
- It will quickly show any changes in the users' needs, circumstances, or opinions, since new feedback will be received whenever such changes occur.

Feedback can be collected by giving users access to special electronic mail address or newsgroups where they can post complaints or suggestions. Of course, user feedback may not always be representative of the majority of the users, and it is thus recommended to supplement it with methods where a representative set of users are actively observed or questioned.

Usability laboratories related testing methods

Thinking aloud

An effective technique during usability testing is to invite users to think aloud about what they are doing. This technique may be the single most valuable usability engineering method. Basically, a thinking-aloud test involves having a test subject use the system while continuously thinking out loud [Lewis, 1982]. By verbalising their thoughts, the test users enable the observers to understand how they view the computer system, and this makes it easy for the observers to identify the users' major misconceptions. One gets a very direct understanding of what parts of the dialogue cause the most problems, because the thinking-aloud method shows how users interpret each individual interface item [Nielsen, 1993].

The main disadvantage of the method is that it does not lend itself very well to most types of performance measurement. On the contrary, its strength is the wealth of quantitative data it can collect from a fairly small number of users [Nielsen, 1993].

The designer or observer should be supportive of the participants, not taking over or giving instructions, but prompting and listening for clues about how well they are dealing with the interface. After a suitable period for accomplishing the task list, the participants may be invited to make general comments or suggestions, or to respond to specific questions. The informal atmosphere of a thinking-aloud session is pleasant, and often leads to many spontaneous suggestions for improvements. In their effort to encourage thinking aloud, some usability laboratories found that having two participants working together produces more talking, as one participant explains procedures and decisions to the other [Shneiderman, 1998].

Coaching method

The coaching method [Mack and Burdett, 1992] is somewhat different from other usability test methods in having an explicit interaction between the test subject and the experimenter (or "coach"). In most other methods, the experimenter tries to interfere as little as possible with the subject's use of the computer, but the coaching method actually involves steering the user in the right direction while using the system.

During the test, the user is allowed to ask any system-related question to an expert coach who will answer to the best of his or her ability. Normally, coaching focuses on the novice user and is aimed at discovering the information needs of such users in order to provide better training and documentation, as well as possibly redesigning the interface to avoid the need for the questions.

Retrospective testing

If a videotape has been made of a user test session, it becomes possible to collect additional information by having the user review the recording [Hewett and Scott, 1987]. The user's comments while reviewing the tape are sometimes more extensive than comments made under the duress of working on the test task, and it is of course possible for the experimenter to stop the tape and question the user in more detail without fearing to interfere with the test, which has essentially been already completed [Nielsen, 1993]. We can still add that retrospective testing is especially valuable in cases where representative test users are difficult to get hold of, since it becomes possible to gain more information from each test user.

Performance measurement

Measurement studies form the basis of much traditional research on human factors and are also important in the usability engineering lifecycle for assessing whether usability goals have been met and for comparing competing products. User performance is almost always measured by having a group of test users perform a predefined set of tasks while collecting time and error data.

When performing a performance study, specific goals should be first, clearly defined, and second, precisely quantified. For example, a goal might be to increase the efficiency of use, and another one the effectiveness. Then, specified tasks should be chosen to be representative of the user's normal task mix. But, in interpreting the results from the measurement study, it is necessary to keep in mind the difference that exists between the principled goal that one is aiming for, and the specific quantification which is used as a proxy of that goal, i.e. the chosen tasks.

Given the quantification of a goal, one still needs to define a method for measuring the users' performance. A way is to bring the users into the laboratory and give them a list of the test tasks to perform. Then one can proceed to the study, and typical quantifiable usability measurements include: the time taken to complete the task, the number of errors, the time spent recovering from errors, the number of different commands utilised by the user, the frequency of use of the help system, etc [Nielsen, 1993].

Logging Actual Use

Logging [Nielsen, 1993] involves having the computer automatically collect statistics about the detailed use of the system (automated data logging), or the observer manually collect predefined behaviors corresponding to the different actions that the user is doing (manual data logging). Automated data loggers are thus programmed to recognise the users' interaction with a program (for example), and create an event for every thing the user asks the program to do. With manual logging, the observer enters predefined codes before a session begins. These codes are aligned to keys on the keyboard and the observer presses those keys as the participant performs the corresponding predefined actions.

Normally, logging is used as a way to collect information about field use of a system after release, but logging can also be used as a supplementary method during user testing to collect more detailed data.

Typically, an interface log will contain statistics about the frequency with which each user has used each feature in the program and the frequency with which various events of interest (such as error messages) have occurred.

Statistics showing the frequency of use of commands and other system features can be used to optimise frequently used features. Features that are not used or that are used very rarely should be investigated to see whether it is possible to improve them or to make them more accessible to the users. It may also be possible to remove such features from the system.

Statistics showing the frequency of various error situations can be used to improve the usability of future releases of the system. If certain errors occur very frequently, one should consider whether it would be possible to redesign the system to avoid these error situations, or at least make them less likely to occur. Also, frequent error messages are certainly candidates for concentrated usability efforts to make them more understandable and constructive.

A final use of logging data may be to study the users' detailed use of an interface to find usability problems that may not be apparent when observing users.

We can add about automated logging that it is one of the best ways to collect usability data since it involves a lot of participants without any expensive and time-consuming tests, even if it also has the inconvenient that it's absolutely impossible to establish whether a user has performed the task he wanted to perform. In fact, interpreting the actions of an individual is extremely difficult, since, for instance, web caches (both client browser caches and Intranet or ISP caches) can intercept requests for web pages, and so these requests might never reach the server and thus not be logged. Another problem is that several users might share the same IP address, making it difficult to distinguish who is requesting what pages.

This major concern, i.e. the fact that logging only shows what the users did, but not why they did it, makes interesting the possibility to combine logging with other methods such as interviews, where users are shown data about their own use of the system and asked to elaborate on whatever interesting phenomena may be evident in the data.

Conclusion

To conclude, it is important to underline the fact that sometimes, inspection methods may find usability problems that are not found by testing methods, and that sometimes they may miss some problems that can be found by testing methods. For example, evaluators are probably likely to overlook usability problems when the system is highly domain-dependent and they have little expertise. Since usability inspection methods and usability testing methods find usability problems overlooked by the other kind of methods, it is recommended that both kinds of methods be used. Furthermore, since different kinds of methods are intended to deal with different parts of the usability engineering lifecycle, and since their advantages and disadvantages can generally partly make up for each other, it becomes even more obvious that those methods are actually used at their best when supplementing each other [Nielsen, 1993].

2.2.4 Usability Data Analysis

Usability testing provides a systematic approach to the evaluation of human-machine interfaces. It comprises a set of techniques which can be implemented to support iterative design decisions. Results of usability testing include diagnosis of design problems, identification of avenues for problem solutions, and the determination of adequate specification compliance for the product being evaluated.

Identify the most difficult tasks

To reach those results, it is necessary to analyse the data. And to begin the analysis, it is a good point to start by identifying the tasks on which users had the most difficulty. This helps to prioritise problems and to stay focused on the worst ones in the beginning. One should thus begin by considering the tasks that did not meet criterion, i.e. the tasks that a predetermined percentage of participants did not complete successfully within the benchmark, if one was determined. Essentially, such tasks represent the vulnerable portion of the product and its support materials [Rubin, 1994].

Identify user errors and difficulties

After the tasks which caused most problems have been highlighted, the errors that caused the incorrect performance should be identified too. An error in this case is defined as any divergence by a user from an expected behavior [Rubin, 1994].

Find the source of the error or difficulty

This is the most difficult and interesting part. Indeed, the source of every error should, if possible, be identified by noting the responsible component or combination of components, or some other cause. This type of analysis is the most labour-intensive from all the post-test ones, but this is also the most important. The objective is to attribute a product-related reason for all user difficulties and/or poor performances.

Of course, some of these sources of error will be obvious, and will suppose very easy to find recommendations, which really flow from the error. But some other determinations will be more challenging. For example, several components may be involved, or the problem might be situated in a message written a few screens before, etc.

To perform this source of error analysis, many areas are normally to be reviewed and considered. Indeed, if the test was pretty important, then one might have one's own notes memory, other's notes and memory, the videotape record, one's understanding of how the product works, and, equally as important, one's understanding of user-centred design. It might be a good idea too to consider the background of the users who made errors, as well as the background of the users who did not.

Chapter 3. An analysis of Usability Data

3.1 Introduction

This chapter's objective is first to make an analysis of two existing products, WebTrends Log Analyser, and Noldus Observer Video Pro, in order to have an idea of the existing products in the usability field, and to get an overview of the good and bad points that should be retained for our system. Second, we intend to perform an investigation into the semantics of typical usability data. This means that we try to discover the similarities that may exist between the different kinds of data, and the requirements these should fulfil in order to be considered as typical (i.e. commonly available). We start by a theoretical analysis, followed by a selection of the data used by WebTrends and Noldus, which aims to keep only the information used by both programs that fulfils the discovered requirements.

3.2 Extant System Analysis of The Noldus Observer Video Pro and The WebTrends Log Analyser

3.2.1 Introduction

The purpose of the extant system analysis is the study of an existing system in order to understand it and to highlight its less usable features, to improve them or to avoid doing these mistakes again. That is why these analyses will focus on the main features of both the Noldus Observer Video Pro and the WebTrends Log Analyser which are systems designed for collecting and analysing usability data. Since our future system will be a visualisation-oriented system, we will also focus on their data visualisation techniques, and on the different ways available for selecting the data to display. They will consist of a usability analysis based a task analysis, of a data analysis and of a critique of the available visualisation tools in those software.

WebTrends Overview

The WebTrends Log Analyser is software that reports on some aspects of a web site's activity, including some information such as who is visiting the site, which pages are the most popular, whether the internet advertising is paying off or what time of day maintenance should be performed. It allows one to learn, for example, which sites and keyword searches have referred the largest number visitors to a particular site.

To collect this information, the WebTrends retrieves data from log files³, analyses the data according to a predetermined set of instructions (a profile) and outputs the data to an comprehensible report in several formats according to another predetermined set of instructions (a memorised report). A profile actually contains information about the log file format, its location, as well as options such as enable DNS resolution or not in the reports (enabling it allows one to report on the web site visitors' domain names, while

³ File in which the web server records some information about visitors' connexion of: visitors' user ID, protocol used, activities at the site ... (See Data analysis for more details)

not doing it means that only the IP address will be available), filter the information one wants to see displayed in the report, etc.

Noldus Overview

Systematic observation is a common approach to behavioral research and an essential component of a wide range of fundamental and applied disciplines. The Observer is an integrated software system for collection, analysis and management of observational data. It supports the entire research process, from the design of observational studies and coding schemes through collection of behavioral data to the analysis and presentation of results. It can be used to record activities, postures, movements, positions, facial expressions, social interactions or any other aspect of human or animal behavior. Instantly available analysis reports supply objective and quantitative data for direct conclusions or further research. During an observation session, key presses are used to log events and the time at which they occur. The Observer time-stamps each entry and checks it against a user-defined configuration.

Why Noldus and WebTrends?

- First, we have recalled in Chapter 2 the existing distinction between manual data logging and automated data logging, where the manual logging involves the observer to manually store the events while the test is being performed, and where automated data loggers are systems programmed to determine the interactions between the user and a system or a website, and to create an event corresponding to any action required from the program or website by the user.

It seems then appropriate to choose a program representing the first kind described above, i.e. systems allowing manual logging, and a program representing the automated data loggers. Now, it appears clearly that this distinction matches exactly the differences of concept existing between both the Observer and the Analyser. Therefore the choice of both those programs seems fully justified from this point of view.

- Furthermore, as described in section 3.2.2 below in the chapter, a major distinction exists between qualitative and quantitative data. Now, for quite simple and obvious reasons that will be explained in Chapter 4, our system will be limited to quantitative data. Therefore, it seems natural, when evaluating other existing systems, to consider systems that also manipulate such data, in order to make the analysis useful and coherent. The fact is that WebTrends and Noldus are both programs intended to collect and study quantitative data, which is one of the reasons that justifies our choice.
- We can make another distinction, at the level of the “location” of the data logging. Indeed, the data used by the log analyser is an example of web-oriented data. The used log files constitute a typical example of automated logged data. Among this data, one generally makes the difference between *Server-side logging* and *Client-side logging* [Hong et al.]:

- Server-side logging: It is a way of quantitatively understanding what numerous people are doing on a web site. It also has the advantage of letting participants work remotely in their own environments. However, this kind of logging makes it extremely difficult to interpret the actions of an individual user.
- Client-side logging⁴: Clients are instrumented with special software so that all transactions are captured. The advantage is that literally everything can be recorded, from low-level events such as keystrokes and mouse clicks to high-level events such as page requests. A drawback is the need for special software to be installed on the client.

WebTrends is obviously a server-side logging tool. But our point is that Noldus can, in a way, be assimilated to a client-side logging tool. Indeed, it is always possible, with this system, to store the different actions performed by a user visiting a web site while observing him. Of course, this would be manual logging, but the fact is that, while it would be feasible on the client side, it would be impossible using Noldus to store any event related to the server side. Therefore, we think here that both systems can be considered as representing respectively client-side logging and server-side logging.

We can thus conclude that these tools are pretty complementary, and supplement themselves in a remarkable way on different levels. Therefore their choice seems fully appropriate, especially in the context of developing a usability data visualisation tool, and furthermore, of investigating the semantics of typical usability data, which will be the object of section 3.2.

⁴ A good example of client-side logging tool is the system developed in UPE by Lisl Blundell, student who was completing a Masters degree on the topic of formal usability testing during our stay in Port-Elizabeth. Basically, the system is an Internet browser looking exactly like a well-known and widespread other browser, but which disposes of data logging features that are completely invisible to the user, data that can serve as input for charts automatically generated by the system.

3.2.2 Usability Analysis

3.2.2.1 Introduction

According to Allan Dix and Janet Finlay [Dix and Finlay, 1998], the usability evaluation of a system has three main objectives:

- Assess the extent of the system's functionality
- Assess the effect of the interface on the user
- Identify the effect of the interface on the user

We can add to those three objectives that the overall goal of usability testing is to identify deficiencies existing in computer-based and electronic equipment and their accompanying support materials. The intent is to ensure the creation of a product that [Rubin, 1994]:

- Is easy to learn and use
- Is satisfying to use
- Provides utility and functionality that are highly valued by the target population.

Now, this usability analysis is aimed at evaluating already existing software, and can therefore be situated, in the usability engineering lifecycle, in the phase corresponding to the competitive analysis. The reason for such an evaluation is that, as explained in Chapter 2, prototyping is a very important phase of the usability process, and already existing products are often the best prototypes to get from a product.

According to [Nielsen, 1993], the already existing products should preferably be evaluated through empirical user tests and heuristically. We will proceed to the only heuristic analysis. This is thus the method we will use in order to evaluate the WebTrends Log Analyser, as well as the Noldus Observer Video Pro.

The result of this analysis can be a list of guidelines for approaches that seem to work and those that should be avoided, as well as several ideas for our new design, based on the existing designs' discovered strengths and weaknesses.

This method, which has been defined in Chapter 2, is used for structuring the critique of a system using a set of relatively simple and general heuristics that can be used to generate ideas while critiquing the system [Shneiderman, 1998]. These heuristics are [Nielsen and Mack, 1994]:

Visibility of system status

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

Match between system and the real world

The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms; it should follow real-world conventions, making information appear in a natural and logical order.

User control and freedom

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.

Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Therefore terminology, abbreviations, formats, colours, capitalisation and so on should all be standardised and respect platform conventions. Consistent sequences of actions should be required in similar situations. Exceptions should be comprehensible and limited in number.

Error prevention and simple error handling

Even better than good error messages is a careful design, which prevents a problem from occurring in the first place. If users make an error, the system should detect the error and offer simple, constructive, and specific instructions for recovery.

Recognition rather than recall

Make objects, actions, and options visible. The user should not be required to remember information from one part of the dialogue to another. Tasks should be arranged so that completion occurs with few actions. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use

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. The user should be able to tailor frequent actions, and to get the information from the display in the form most convenient for the task on which they are working

Aesthetic and minimalist design

Dialogues should not contain information that 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.

Help user recognise, diagnose and recover from errors

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

Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

In the following sections, we try to assess each of these heuristics for the main tasks described in the following task analysis. Then we will discuss the problem of data visualisation in the process of analysing data collected by WebTrends and Noldus.

3.2.2.2 Task Analysis

Usually, as said in Chapter 2, the objective of a task analysis is to create a list of all the things users want to accomplish with the system, the information they need, etc. This analysis is thus supposed to be based on the users' desires. However, the objective is not to found our evaluation on the users' demands, but rather to describe the different tasks available for the users in the existing systems. This analysis is thus based on software exploration and tests, and not on user interviews. The outcome will be used in section 3.1.2.3, i.e. in the heuristic analysis, in order to assess the extent in which the discovered tasks satisfy the above-defined heuristics.

According to Shneiderman [Shneiderman, 1998], "*High-level task actions can be decomposed into multiple middle-level task actions that can be further refined into atomic actions that the user executes with a single-command, menu selection, and so on*". In the following, we respect this structure, and thus after identifying the main tasks of both the Observer Video Pro and the Log Analyser, we try to refine them in smaller actions.

WebTrends

Main tasks

The system can be divided into five main tasks:

- 1. Design a profile**
- 2. Save report settings**
- 3. Customise reports**
- 4. Run a report**
- 5. Schedule reports**

Before creating a report on the web site's activities, the first step is to design a profile that identifies the location of the log files and how the data should be processed. The second and third steps are saving settings for this report and if necessary, customising it. After that, it is possible to run the report or to schedule it at a given date. These five tasks can be illustrated by an Extant Task Model as given below. This Extant Task Model is limited to the 3 first levels, and thus describes the main tasks. A more detailed model is included in Appendix A.

Extant Task Model

1. Design a profile

- 1.1 Create a new profile
 - 1.1.1 Create new profile
 - 1.1.2 Edit new profile
- 1.2 Modify an existing profile
 - 1.2.1 Select profile
 - 1.2.2 Edit profile
- 1.3 Copy an existing profile
 - 1.3.1 Select profile
 - 1.3.2 Copy profile
- 1.4 Delete an existing profile
 - 1.4.1 Select profile
 - 1.4.2 Delete profile

2. Save report settings

- 2.1 Select a profile
- 2.2 Select to create report
- 2.3 Select the data range for the report
- 2.4 Select a file format for the report
- 2.5 Select method and destination for saving report
 - 2.5.1 Select method
 - 2.5.2 Select destination and name
- 2.6 Specify a style for the report
 - 2.6.1 Enter report title
 - 2.6.2 Select language
 - 2.6.3 Select style
 - 2.6.4 Select whether to include help cards in the report
- 2.7 Specify the content of the report
- 2.8 Memorise the report settings
 - 2.8.1 Enter a name for the report settings
 - 2.8.2 Confirm memorise the report settings
- 2.9 Cancel create report

3. Customise reports

- 3.1 Customise styles
 - 3.1.1 Create a new style
 - 3.1.2 Modify an existing style
 - 3.1.3 Delete an existing style
 - 3.1.4 Rename an existing style
- 3.2 Customise language
 - 3.2.1 Select language
 - 3.2.2 Confirm customise language

4. Run a report

4.1 Select profile

4.2 Create report

4.2.1 Select a memorised report

4.2.2 Run report

5. Schedule reports

5.1 Schedule events

5.1.1 Schedule an event

5.1.2 Edit a scheduled event

5.1.3 Delete scheduled event

5.2 View history of scheduled events at a given date

5.2.1 Enter data

5.2.2 View events

5.3 View the system's performance by event

5.3.1 Select day

5.3.2 View performances

5.4 Select options on how scheduled events are processed

5.5 Activate remote scheduling centre

5.5.1 Specify a remote reporting server

5.5.2 Start server

5.5.3 Configure user account for remote scheduling

5.6 Set up scheduler to run as a service

5.6.1 Enter username

5.6.2 Enter password

5.6.3 Start service

5.6.4 Configure the service start-up

5.6.5 Add privileges to NT account

Noldus

Main tasks

The Observer has a modular structure. The program tasks have been divided over multiple subprograms or modules. The Base Package includes the following modules: Project Manager, Configuration Designer, Event Recorder and five Data Analysis Procedures. The Project Manager is the centre of all modules, and allows the user to launch other program modules, but it does not really constitute a main task, though the other modules do. The design of the observational study is specified in the Configuration Designer. For each observational project a different configuration can be created. In this Configuration Module is actually defined what one wants to observe and how one wishes to record it. In the Event Recorder are the actual observations performed, the data being automatically saved in observational data files. In the Data Analysis, the data can be analysed in various ways, thanks to five powerful analytical procedures. Though all these procedures are useful, we have decided to do the Task Analysis of the only Time-Event Plot, because of the similarity of the interfaces, and to avoid getting redundant results.

Basically, the system can be divided into three main tasks:

1. **Configure Design (Configuration Designer)**
2. **Collect Data (Event Recorder)**
3. **Analyse Data (Time-Event Plot)**

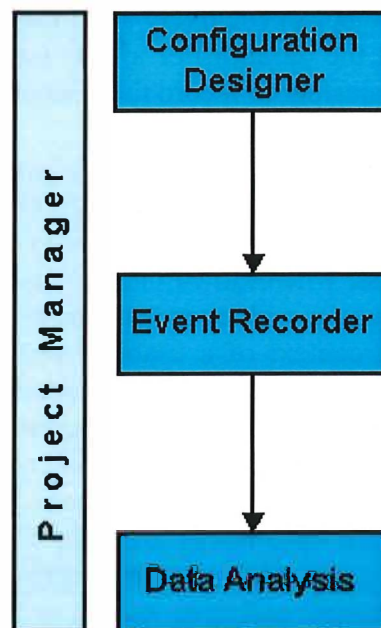


Figure 2: Modules of the Observer

An Extant Task Model as given below can illustrate these three tasks. This Extant Task Model is limited to the first levels, and so describes the main tasks. A more detailed model is included in Appendix A.

Vocabulary

Here are the definitions of some words used below:

- *The Configuration Module:*
 - Event: Combination of subject, behavioral element and modifiers.
 - Independent variables are used for circumstances that do not change within an observation, but do change between observations, like the name of the subject, his or her gender, etc. Actually, they are descriptors of observations of which the values are constant during an observation (it is possible to define up to 40 independent variables).
 - A subject, in the Noldus terminology, is an individual under observation. To each subject is associated a description and a keyboard code.
 - The behavioral elements are the basic items scored during an observation session. To each behavioral element is associated a description and a keyboard code. A behavior can be defined as an event (an event is an element of relatively short duration, for which only the occurrence matters, not the duration) or a state (states have a relatively long duration, and it is interesting to record their beginning as well as their end).
 - Behavioral Classes: Behaviors can be grouped in one or more classes. A class is a group of mutually exclusive categories. This means that in a class, only one behavioral state can be active at a time. Consequently, a state in a class, when turned on, automatically turns off the previous state in the same class. In each class, one behavior can serve as the default state. Example: in the context of a usability test where five behavioral elements are defined (e.g. mouse clicked, key pressed, inactive, read and look away), we could consider two behavioral classes, which could be, for instance, hands and look. Hands would then contain the behavioral elements mouse clicked, key pressed and inactive and look the two other ones. Thus, in the first class, the active state would be either inactive, or mouse clicked or key pressed, and only one of these. In the other class, i.e. look, we could consider that the user is either reading or looking away. Basically, classes do not really bring much information, particularly in the field of usability testing, their main advantage actually being that when one element is activated, then the active one is automatically ended, which allows one to save coding time.
 - Modifiers can give more information on an observed behavior. They can be the direction, location, amount, intensity, etc. They can be used to refine the information, and to indicate the limits or the scope of a behavior. Each behavioral element can have up to two modifiers.
 - Modifiers Classes: Modifiers can be grouped in classes. For example, a class direction with elements up, down, left and right.

- A channel is a unique combination of a subject and a behavioral class. The behavioral states in a channel are always mutually exclusive, and their duration's add up to 100% of the observation time. For example, in a study of parent-child interactions with three subjects (Father, Mother and Child) and two behavioral classes (Behavior and Location), there are six possible channels, of which e.g. four are used: Child*Behavior, Child*Location, Father*Behavior and Mother*Behavior. Channels are useful to structure the coding system, and to focus on what is relevant to the research question. However, it is also possible to enable scoring and analysis of events outside channels. Furthermore, channels are not really interesting in the field of usability data, where most of the observations make intervene only one living subject at a time, another subjects sometimes being the tested or used device.
 - Review Configuration: The configuration can be presented in a tabular format for easy on-screen viewing.
 - Check Configuration: This function checks the completeness and consistency of all definitions, to prevent surprises during data collection.
- *The Observation Module:*
 - Customising: The observation views can be defined according to one's personal taste. A customisation includes settings that determine how the program behaves during an event-recording session, as well as screen-layout options, event timing, data entry, data storage and auditory feedback signals.
 - The event-recording screen has a number of windows, each of which can be open or closed, positioned and resized to one's liking:
 - Event Log - The Observer's metaphor of a paper notepad. It shows the chronological event log, i.e. all events and the time at which they are scored. Properties include time format, columns to be displayed, and column width. An example is shown on Figure 3.

RECORD	TIME	ACTOR	BEHAVIOR	COMMENT
8	00:00:10.44	Derek	Walk	
9	00:00:16.92	Derek	Run	
10	00:00:21.36	Derek	Visual contact	
11	00:00:26.48	Edward	Play	
12	00:00:31.32	Jerry	Play	
13	00:00:35.00	Jerry	Contact	
14	00:00:38.96	Jerry	Exchange	
15	00:01:00.32	Derek	Push or Pull	
16	00:01:06.80	Jerry	Outside	
17	00:01:09.72	Edward	Outside	
18	00:01:13.08	Edward	Run	
19	00:01:16.12	Edward	Play	
20	00:01:38.00	Jerry	Element	

Figure 3: Event Log

- Timers - Start time, current time, end time, elapsed time, observed time, maximum time, remaining time, time to next sample. These clocks, which are constantly updated during data collection, are all available, and as many of them as desired can be displayed or hidden at the same time. It is also possible to set the time resolution.

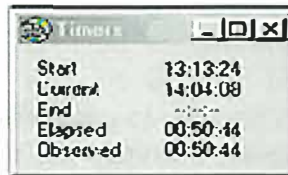


Figure 4: Timers

- Channels - For each channel, the current state is shown, allowing a quick check if it corresponds with the current situation.

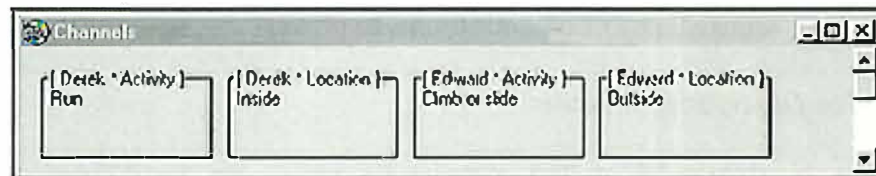


Figure 5: Channels

- Checklist - A list of all subjects that have to be scored.
- Codes - A customisable table with the valid codes at each step in the data-entry process. It helps to memorise how the keys of the keyboard have been defined. It can also be used for direct data entry with a mouse or pen stylus.

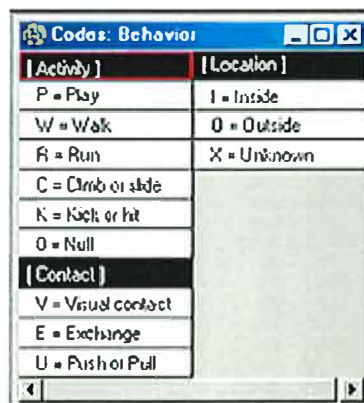


Figure 6: Behavior codes

- Toolbar - A floating box with (customisable) icons for use as shortcuts to frequently used actions.



Figure 7: Toolbar

- Notepad - Type notes to be stored with the observational data. Each note is time stamped.
 - System status - Displays several system variables, including file names, amount of free disk space, configuration settings, etc.
- Edit the event log: It is possible to edit previously entered codes during an observation session. In the on-line Event Editor it is possible to scroll back in time and correct coding errors in earlier data records
- Suspend Observation: During a session, an observation can be paused at any time. As long as the observation is suspended, the elapsed time continues to run, but the observed time is not affected.

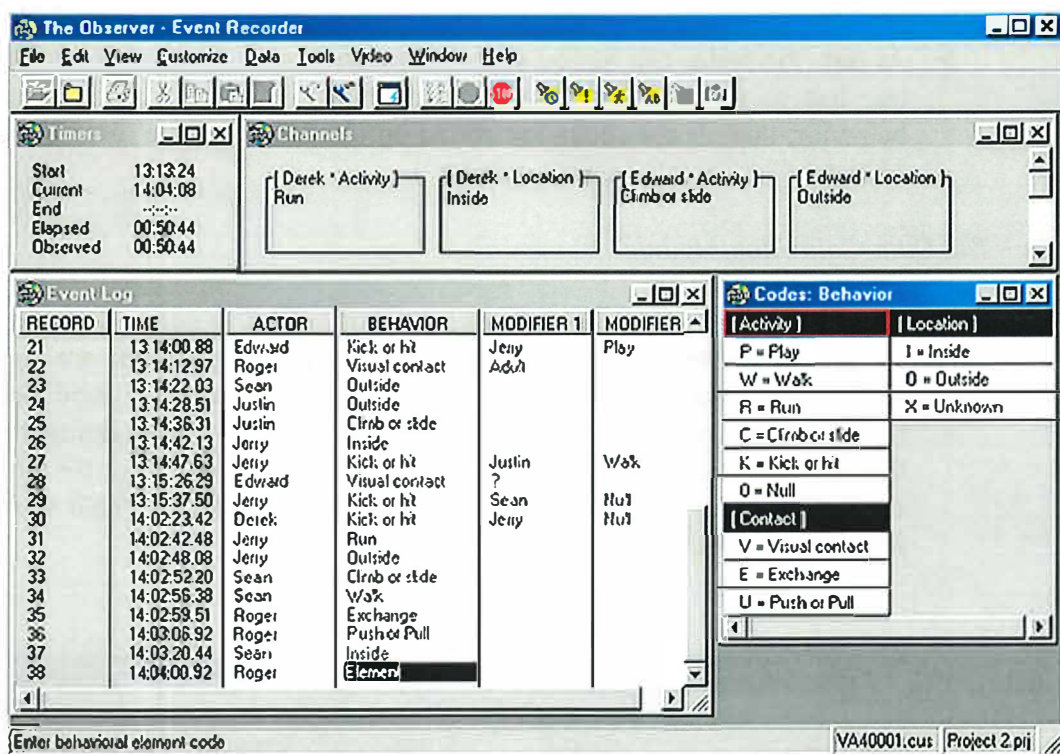


Figure 8: The Observation Module during a scoring session

Some of the above-mentioned concepts are visible on Figure 8:

- The timers are situated in the upper left of the window. There, the start time, the current time, the end time, which is empty since the observation is still running, the elapsed time, and the observed time are displayed. The difference between the elapsed and observed times is that it is possible to pause the observation, which pauses the observed time, but not the elapsed time.
 - The event log is just under them and fills both the bottom and left part of the screen. There, one can see different columns, the first one's being called record, and obviously containing the number of order attributed to each stored event. The time column contains the time to which the stored event started, the actor column contains the name of the user responsible for or involved in the stored event, the behavior column gives the name of the behavior corresponding to the stored event, and both modifier columns contain data refining the behavior.
 - The channels are shown as well, with for each of them, the name of the subject and the name of the class composing the channel, and the name of the currently active behavioral element.
 - At last, the codes can be found under the channels. These codes are the keys one has to press in order to store an event. Each code corresponds to a behavior, and all the codes are sorted on classes (here, the three classes are: "Activity", "Contact" and "Location").
- *The Data Collection Module:*
 - Data Selection: Data can be selected (filtered) and sorted for analysis at several hierarchical levels. The following selection filters are available: independent variables, observations, subjects, behaviors, modifiers, and intervals (the section of data to include in the analysis can be defined through the start criterion and the end criterion, which can be either time-based or event-based, e.g. from time = 30 to event John looks). The data selection is presented in grid views, as show in Figure 9.

Behavioral Class	Analysis Group	Behaviors	Color	Modifier 1	Modifier 2
1	Walking or Running	Walking, Running	...	Place	<none>
2	Sitting	Sitting	...	Place	<none>
3	Standing	Standing	...	Place	<none>
4	Unspecified	Unspecified	...	Place	<none>
5 *					

Navigation: Independent Variables / Observations / Subjects / Behaviors / Modifiers / Intervals

Figure 9: Data Selection

Extant Task Model

1. Design a configuration

- 1.1 Specify the Data Collection Method
- 1.2 Specify the Timing Method
- 1.3 Specify Independent Variables
 - 1.3.1 Add an independent variable
 - 1.3.2 Remove an independent variable
 - 1.3.3 Replace an independent variable
- 1.4 Specify Subjects
 - 1.4.1 Add a subject
 - 1.4.2 Remove a subject
 - 1.4.3 Replace a subject
- 1.5 Specify Behavioral Classes
 - 1.5.1 Add a behavioral class
 - 1.5.2 Remove a behavioral class
 - 1.5.3 Replace a behavioral class
 - 1.5.4 Specify Behavioral Elements
 - 1.5.4.1 Select a behavioral class
 - 1.5.4.2 View behavioral elements
 - 1.5.4.3 Modify behavioral elements
 - 1.5.4.3.1 Add a behavioral element
 - 1.5.4.3.2 Remove a behavioral element
 - 1.5.4.3.3 Replace a behavioral element
- 1.6 Specify Modifier Classes
 - 1.6.1 Add a modifier class
 - 1.6.2 Remove a modifier class
 - 1.6.3 Replace a modifier class
 - 1.6.4 Specify Modifier Elements
 - 1.6.4.1 Add a modifier element
 - 1.6.4.2 Remove a modifier element
 - 1.6.4.3 Replace a modifier element
- 1.7 Specify Channels
 - 1.7.1 Add a channel
 - 1.7.2 Remove a channel
 - 1.7.3 Replace a channel
- 1.8 Review the configuration
- 1.9 Test the configuration
 - 1.9.1 View the result

2. Collect Data

- 2.1 Customize the Event Recorder
 - 2.1.1 Specify Operation
 - 2.1.2 Customize windows
- 2.2 Define independent variables
- 2.3 Perform an observation session
 - 2.3.1 Start an observation
 - 2.3.2 Edit observational data

- 2.3.3 Suspend the observation
- 2.3.4 Log events
- 2.3.5 Scan channels (States or Events)

3. Data Analysis using a Time-Event Plot

- 2.4 Fill cells in the worksheet
- 2.5 View graph
- 2.6 Zoom in
- 2.7 Zoom out
- 2.8 View the full view
- 2.9 Define a time-based filter
 - 2.9.1 Select the point from where to plot the graph
 - 2.9.2 Select the point to where to plot the graph
- 2.10 Plot a channel
 - 2.10.1 Select a channel
 - 2.10.2 Select a subject
- 2.11 Specify the grid
- 2.12 Specify the legend
- 2.13 Specify the title
- 2.14 Select the trace styler

3.2.2.3 Usability Analysis

Now that we have determined the different tasks available in both software WebTrends and Noldus, and that we have expressed them under the shape of a task analysis that has allowed us to discover the smaller actions composing them, we can proceed to their heuristic evaluations. As said in Chapter 2, the outcome of such an evaluation should be a list of usability problems, each of them being annotated with references to the violated heuristics. But, one of the objectives of competitive testing is to find out some guidelines for approaches that seem to work. Therefore, we will present the results of our evaluation under the shape of a table, with, for each heuristic, the positive points we have discovered in the software, as well as the negative ones, each of them being related to one of the above-mentioned tasks.

We do not include screenshots of the evaluated programs since this analysis is intended to find out general qualities and defaults of usability-related programs that might inspire us for ours, but not to particularly emphasize and insist on WebTrends or Noldus problems.

WebTrends

Results of the Evaluation

<i>Heuristics</i>	<i>Evaluation</i>	
	<i>Positive</i>	<i>Negative</i>
Visibility of system status	<ul style="list-style-type: none"> - System status visible through tabs (Task 1, 2 and 5) - Graphical overview of options selected (Task 3) 	<ul style="list-style-type: none"> - No feedback given on the progress of the task (Task 3)
Match between system and the real world	<ul style="list-style-type: none"> - Right terms according to the web language (Task 1) - Graphical overview of options selected (Task 3) 	<ul style="list-style-type: none"> - Technical terms such as URL, DNS, Database (Task 1) - Sequence of action not logical (Task 2)

<p>User control and freedom</p>	<ul style="list-style-type: none"> - Possibility of coming back on a tab (Task 1, 2 and 5) - Next and back buttons (Task 3) - Possibility of redesigning a profile (Task 1) - Possibility of redesigning a memorised report (Task 2) - Possibility of re-customising a report (Task 3) - Possibility of reediting a scheduled report (Task 5) 	
<p>Consistency and standards</p>	<ul style="list-style-type: none"> - Similar from a task to another - Similar to the tools of the platform used (i.e. Windows) 	
<p>Error prevention</p>	<ul style="list-style-type: none"> - Examples, advice explaining what to do and why (Task 1) - Only combo boxes and check boxes (Task 2) - No errors possible (Task 3 and 4) 	
<p>Recognition rather than recall</p>	<ul style="list-style-type: none"> - Default options and text field contents - Examples (Task 1) 	
<p>Flexibility and efficiency of use</p>		<ul style="list-style-type: none"> - No accelerators (no need)

Aesthetic and minimalist design	- (Task 1, 2, 3 and 4)	- Too much things in a window (Task 5)
Help user recognise, diagnose and recover from errors	- No errors possible (Task 2, 3 and 5)	- No errors detected. All checks made at task 4 (Task 1) - No help to find where an error has been made on tasks 1 and 2 (Task 4)
Help and documentation	- Contextual help available, lots of examples - Good manual providing screenshots for all possible windows - Very clear and easy tutorial available at program start	

Comment

Basically, we can say that WebTrends is usable. Though the terms used are generally related the web environment which of course demands from the users a certain knowledge of this field, this requirement can be considered as acceptable since WebTrends is not supposed to be used by novice users. Different other positive points are present, especially in the matters of user control, which is very good, consistency and standards (these issues being pretty well addressed), and error prevention, through numerous examples and advice, and even the complete absence of possibility to make any error in some tasks. Unfortunately this not the case for all of them, but we can notice anyway that a global effort has been furnished in order to make this software as usable as possible. Of course it is far from being perfect, but the positive points are much more numerous than the negative ones. We will thus try to keep in mind most of those issues when designing the future system.

Noldus

Results of the evaluation

<i>Heuristics</i>	<i>Evaluation</i>	
	<i>Positive</i>	<i>Negative</i>
Visibility of system status	<ul style="list-style-type: none"> - Tables showing all the scored events (visual feedback during coding), the different timers, ... (task 2) 	<ul style="list-style-type: none"> - Feedback about the quality of the configuration available only at the end of the configuration (task 1)
Match between system and the real world	<ul style="list-style-type: none"> - No technical terms, but a few problems of semantics (task 1, 2), such as for the different sampling methods - Zoom in and out (task 3) - Possibility to filter details (task 3) 	<ul style="list-style-type: none"> - Zoom not intuitive at all (task 3)
User control and freedom	<ul style="list-style-type: none"> - Undo, Redo available (task 1, 2, 3) - The Event Recorder is fully customisable (task 2) - Video playback available (task 2) 	
Consistency and standards	<ul style="list-style-type: none"> - Much consistency 	<ul style="list-style-type: none"> - In all menus, the Ok and Cancel button are above the Add button and this leads to errors (task 1)

<p>Error prevention</p>	<ul style="list-style-type: none"> - Not a lot of possible errors (task 2, 3) - Some fields give only the possibility to enter numbers (task 1, 2, 3) 	<ul style="list-style-type: none"> - Lots of error messages guiding the user, but often displayed at the end of the task, i.e. when the error already occurred (task 1) - The Add button stays editable though all fields that should be filled aren't, and clicking on it then makes an error message appear... (task 1) - No confirmation before removing items in the different menus (task 1) - No distinction between optional or compulsory fields (task 1)
<p>Recognition rather than recall</p>	<ul style="list-style-type: none"> - Tables available to remember the codes (task 2) 	<ul style="list-style-type: none"> - No tooltip texts on the icons! (task 1, 2, 3)
<p>Flexibility and efficiency of use</p>	<ul style="list-style-type: none"> - Different possible ways to score the events (task 2) - Accelerators (task 2) - Different possible ways to collect data (from video tape, live, ...) (task 2) 	<ul style="list-style-type: none"> - No possible interaction in the analysers: indeed, as said above, the data selection is possible only before starting the analysis

Aesthetic and minimalist design	<ul style="list-style-type: none"> - The menus are not overloaded 	<ul style="list-style-type: none"> - Only 10 colours available in the Time-Event plot (task 3) - The different analysers aren't really aesthetic and easily comprehensible
Help user recognize, diagnose and recover from errors	<ul style="list-style-type: none"> - Explicit error messages (task 1) 	
Help and documentation	<ul style="list-style-type: none"> - Help available everywhere (task 1, 2, 3) 	<ul style="list-style-type: none"> - No screenshots (task 1, 2, 3)

Comment

Compared with WebTrends, Noldus is not as usable as its web-based equivalent. Indeed, first of all, the interfaces are generally quite different from what we are used to see. They contain many small imperfections and do not respect all the usual standards. Furthermore, the programme, which is based on principles and ideas that are not easy to understand, does not really speak the user's language, but uses its own terminology which is definitely not immediately understandable, and requires some learning. The absence of screenshots in the help for instance makes it very complicated to interpret the meaning of the different menus and options. Another negative example is the message describing all the errors made during the configuration. The only existence of the possibility to make errors is already a negative point, and the fact of being informed of them only at the end of a task makes this negative point even worse. Many imperfections should be addressed to facilitate and make more comfortable the use of this programme. There are some other critiques that can be made about Noldus, but we will not go further in this direction as far as all of them lead to the same conclusion, which is that on a usability point of view, this programme might be much better. All those errors constitute therefore good examples of problems that should be avoided for the conception of our system. Of course there are not only negative points, and a good one is the high customizing capacity of the Event Recorder. But most of the positive points can be considered as necessary characteristics. As a conclusion, we will say that Noldus is not as usable as it should be, and that a lot of improvements should be done. Nevertheless, we still consider that this tool, since it has a lot of possibilities, has a high potential, and is, from that perspective, a very interesting program.

3.2.2.4 Visualisation evaluation

WebTrends

The Charts

2D column chart

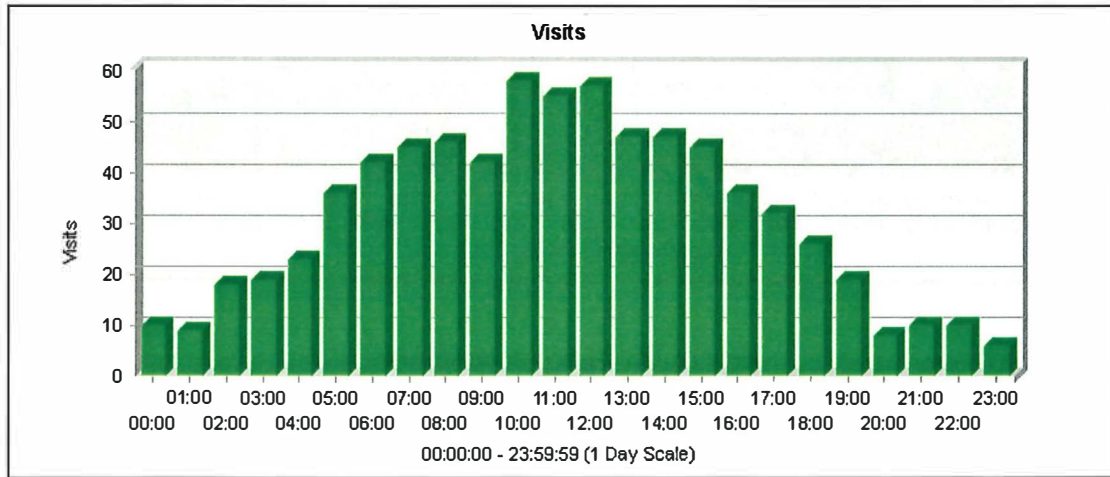


Figure 10: 2D column chart

Critique:

On Figure 10 is displayed a 2D column chart representing the overall number of visits to a web site during one day, hour per hour. Each column represents an hour, and the column's height corresponds to the number of visits. The data displayed on this graph is very clear. We can see that the moment of the day during which the number of visits is the highest is the noontime, while the moment when this number is the lowest is around midnight. This data is very clearly presented and is thus easily interpretable. However, and this remark is true for all the available graphs under WebTrends, there is no way to interact with the graph in order to zoom, to select a specific hour etc.

3D column chart

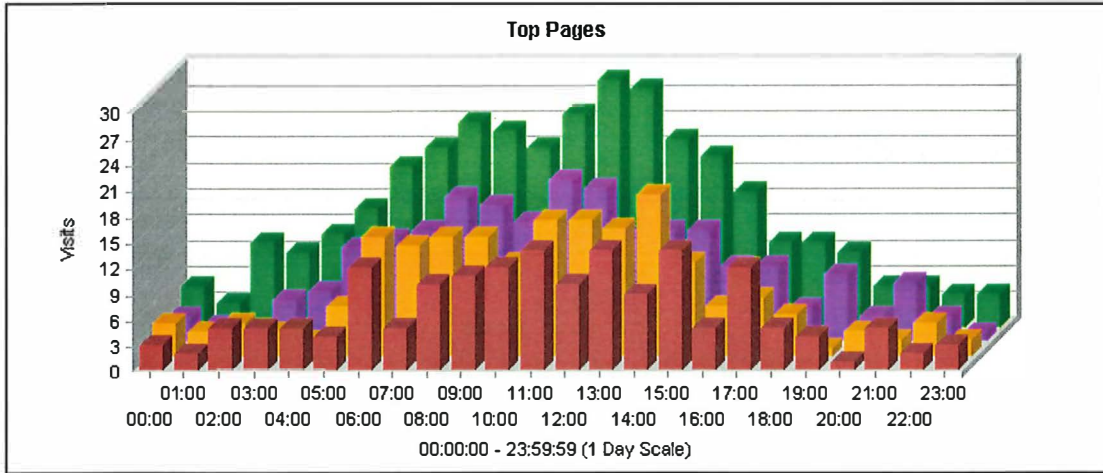


Figure 11: 3D column chart

Critique:

This chart presented on Figure 11, which represents data related to the most popular web pages on the studied web site, shows how often they are visited, and displays the average length of time the page was viewed. A very obvious problem here is the too high number of pages showed at the same time. Indeed, the graph has a depth of four columns, which is a lot, since it is impossible to move it in anyway. Furthermore, the columns are very close to each other, which overloads the graph and thus does not facilitate visualisation. Therefore, the risk is big to have completely masked columns, which is the case here for some of the columns.

Stacked column chart

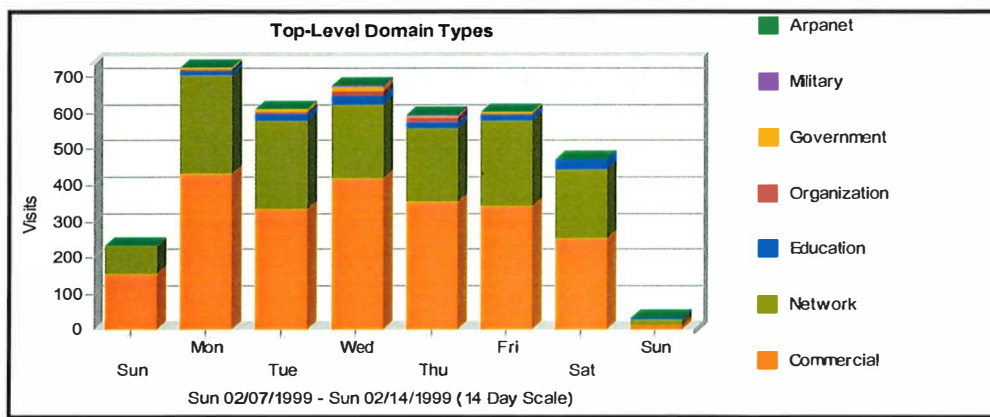


Figure 12: Stacked column chart

Critique:

The chart on Figure 12 represents the number of visits per day of the week, each colour representing a domain type. Though this chart gives a quick overview of the general trends in this field, again, the lack of interactivity and more particularly the impossibility to zoom makes it impossible to distinguish the colours displayed on the top of the different columns. Another point is the impossibility to click on the different coloured areas in order to get details about the accurate number of visits. This thus means that the user desiring to get more detailed information is supposed to check the textual data, when this quite simple information (one number) could have been easily displayed on demand. We will definitely pay attention to this point in the design of our system.

Area Chart

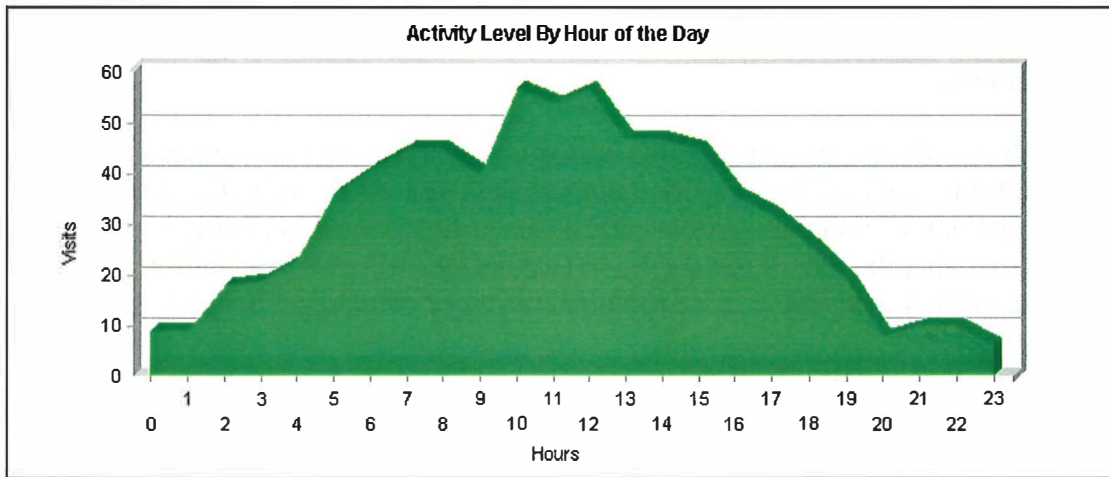


Figure 13: Area chart

Critique:

This graph shows activity for each hour of the day. As stated in [Darville and Van Espen, 2000], "an area chart is suitable at underlining the trends of information along a time period". However, it is yet quite redundant with the 2D column chart presented on the above Figure 13.

Pie chart

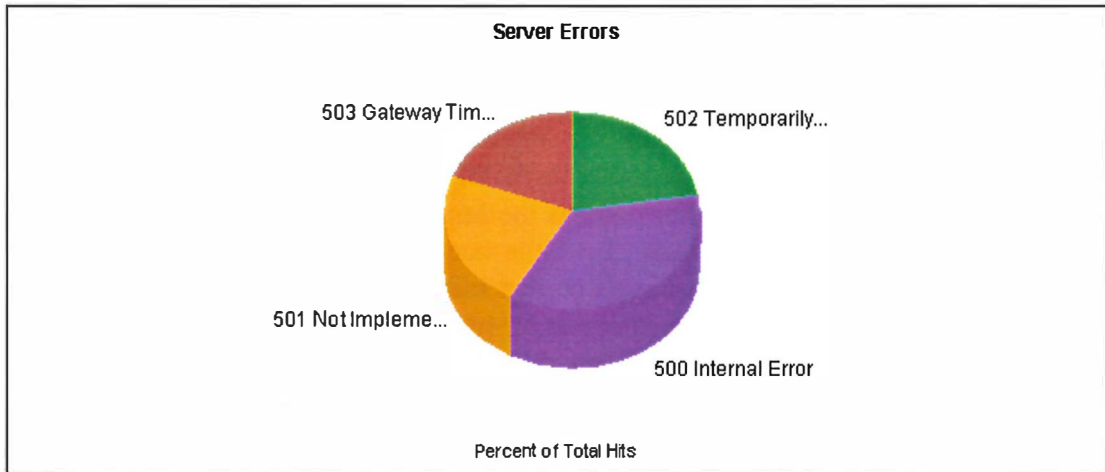


Figure 14: Pie chart

Critique:

This graph lists the errors that occurred on the server. First, we can immediately notice the difficulty to estimate the relative values represented by the different pieces of the chart. Actually, this impression only confirms the critique stated in [Tuft, 1983], which basically says that the use of pie charts, in account of their poor readability, should be avoided as much as possible. For information, the corresponding values are:

- 500 internal error: 35.48 %,
- 501 not implemented: 22.58 %
- 502 temporary overloaded: 22.58 %
- 503 gateway timeout: 19.35 %.

Conclusion

The report created by the WebTrends Log Analyser provides simple and exact information about analysed log files. This report contains of course textual descriptions, data tables and helps on how to interpret this data, but also provides a lot of ways to visualise the data including well-known 2 and 3D charts such as column charts, stacked column charts, area charts, and pie charts. However, there is a complete lack of interactivity among these graphs. Although the report can be run in a browser, it is for example impossible to change view or zoom in when a chart is not visible enough. It is furthermore impossible to get more information by clicking on a chart. Log analyser provides only images that can be easily inserted into a Word, Excel or HTML document (see Figures 10-14 for examples).

Noldus

The charts

The Observer contains five data analysis procedures. One of those, called the Reliability Analysis, was designed to measure the level of agreement between pairs of evaluators, or between pairs of data files scored by one observer. The context of this project makes this tool irrelevant, and so it will not be treated in the following parts of this report. The four other tools will be described in more detail below.

Time-Event Table (Textual View)

A time-event table is a sequential, chronological listing of all recorded events, sorted on columns for the various subjects and classes of behavior. Its main features are:

- Separate column for each subject/class combination
- Clearly visualises the change of states and the occurrences of events in the course of time
- Use filters to select the data to be visualised, especially a time window.

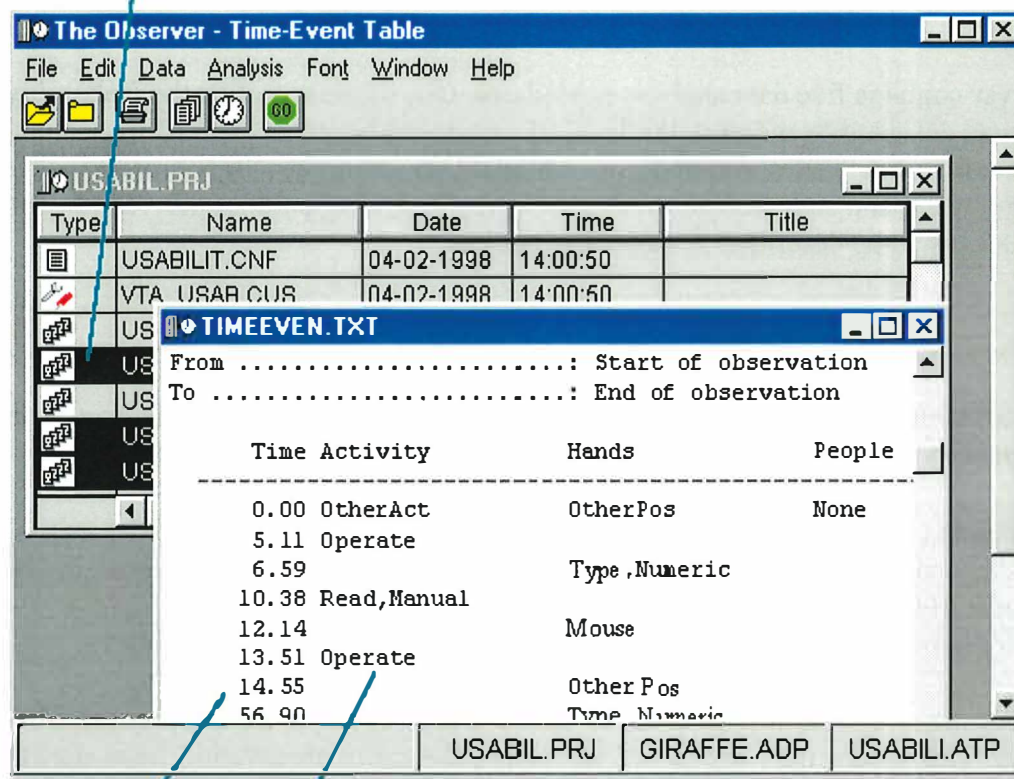
Through this menu option, it is possible to define a time-based or an event-based filter for analysis. This means that, instead of processing the entire observation from start to end, it is possible to select events, based on whether they belong to the time-interval defined. It might be useful, e.g. to visualise the part of the observation between five and ten minutes elapsed time, or to know what happened between the events *John – Keypress* and *John – mouseclick*.

The time window can also be used to calculate statistics for repetitive intervals. This is useful, e.g. to know the frequency of a particular behavior for successive periods of five minutes, or to know the frequency of behaviors that occur for each interval marked by the events *John – Keypress* and *John – mouseclick*. This time-based filter is very useful and its characteristics should, in our opinion, be present in the new system.

However, the available data selection tool is rather heavy to handle, since it is necessary, every time one wants to change the selected period of time or events to display, to open the window allowing this selection, to select the desired data, to click ok, and only then is it possible to view the new data. This lack of interactivity is quite a problem and we will address this point during the design of our system.

Figure 15 below is represented an example of a time-event table, the interesting part being on the foreground. We can see from left to right the main columns available: time, activity, and hands. The time indicates of course the moment at which the event happened or started, depending on its nature, and the activity describes the event.

the Project Manager keeps your files together



accurate timing

overview of scored behaviors

Figure 15: Example of Time Event Table

Critique:

This table is thus, as described above, a sequential, chronological listing of all recorded events, sorted on columns for the various subjects and classes of behavior. The question that we will try to answer for this table is about the deductions that can be made from the data displayed in this table on a usability point of view.

First of all, this table is presented under a text format, which is not immediately understandable, though the division in columns is rather appropriate and expressive. Second, some measurements can be deduced for each observation separately:

- If the program is configured in the appropriate way, it is possible to know if the task has been completed or not, and thus to know if the completion rate for this observation is 100% or not.
- Still in the case where the configuration is correct, it is possible to see all the errors that have occurred or been committed during the observation.
- The task time or at least the duration of the observation is of course available.

- This tool also provides the possibility to follow all the actions performed by the observed person during the whole duration of the observation, which is very interesting on a usability point of view.

However, this table shows only basic data, in a textual way, and though some information can be deduced from it, this tool is certainly not the most appropriate one for analysing this data.

As a conclusion, all we can say is that this tool is not particularly original. The concept of a table seems appropriate, and should be present in the future system, but only as a complement for other visualisation techniques.

Time-Event Plot (Graphical view)

This is a graph in which observational data are plotted horizontally against a horizontal time axis; in other words, a time-event plot is a time-event table in a graphical layout. A time-event plot gives a quick overview of what happened and when. It gives an impression of whether behaviors are evenly or irregularly spread over time, as well as whether there are large variations in the duration of behaviors.

The plot consists of a user-defined number of traces. Each trace either represents:

- The occurrence of one particular event or state. In this case the occurrences of states can be visualised either as a solid bar or as a line graph.
- The occurrences in a single channel or any combination of an actor and a class of behavioral elements. In this case, blocks of different colours or patterns are generated for each behavioral element. If a single-actor configuration is being used, it is possible to plot all occurrences of behavioral elements belonging to the same class in this way.

A time-event plot can be used to get a quick impression of the nature of data, e.g.:

- Whether behaviors are evenly or irregularly spread over the observation time.
- The variation in the duration of behaviors.
- The relationship between events in different behavioral classes.
- The relationship between events related to different subjects.
- The sequential relationship of events.

As soon as a project is open, a worksheet is displayed. In each row of the worksheet a trace can be defined. A trace can display the occurrence of one single event string (i.e. actor-behavior-modifier 1-modifier 2 combination), a specific subject-behavioral class combination or a single behavior irrespective of who was the subject. There is a maximum of 32 traces per plot. Each element of a trace is defined in a cell. The following seven columns can appear in the worksheet:

- **Trace** - The cells of this column are numbered from 1 to 32. This is the number of a trace. This number is also displayed in the graph and the legend.
- **Observation** - In this column it is possible to select the observational data file from which to plot the data. The drop-down list of an observation cell consists of the observational data files belonging to the project.
- **Subject** - If a multiple-actor configuration is being used, it is possible to define a subject in this column. This column does not exist in case of a single-actor configuration. The drop-down list of a subject cell consists of the subjects defined in the configuration on which the project is based.
- **Behavior** - In this column it is possible to select the desired behavioral elements or classes. The drop-down list of a behavior cell consists of the behavioral elements and classes defined in the configuration on which the project is based.
- **Actor/Receiver (A/R)** - In this column it is possible to select to plot either the events in which the subject in the subject column is the actor or those in which the subject is the receiver.
- **Modifier 1** - A cell in the modifier 1 column can only be accessed if a single behavior is selected instead of an entire behavioral class in the behavior column, and if the selected behavioral element uses at least one modifier.
- **Modifier 2** - If the selected behavioral element is using two modifiers, it is then possible to select a modifier 2.

Options are available to zoom in on a part of a time-event plot, to zoom out again, or to restore full view of the plot. If a time window is specified, the plot will be printed zoomed in on the time window.

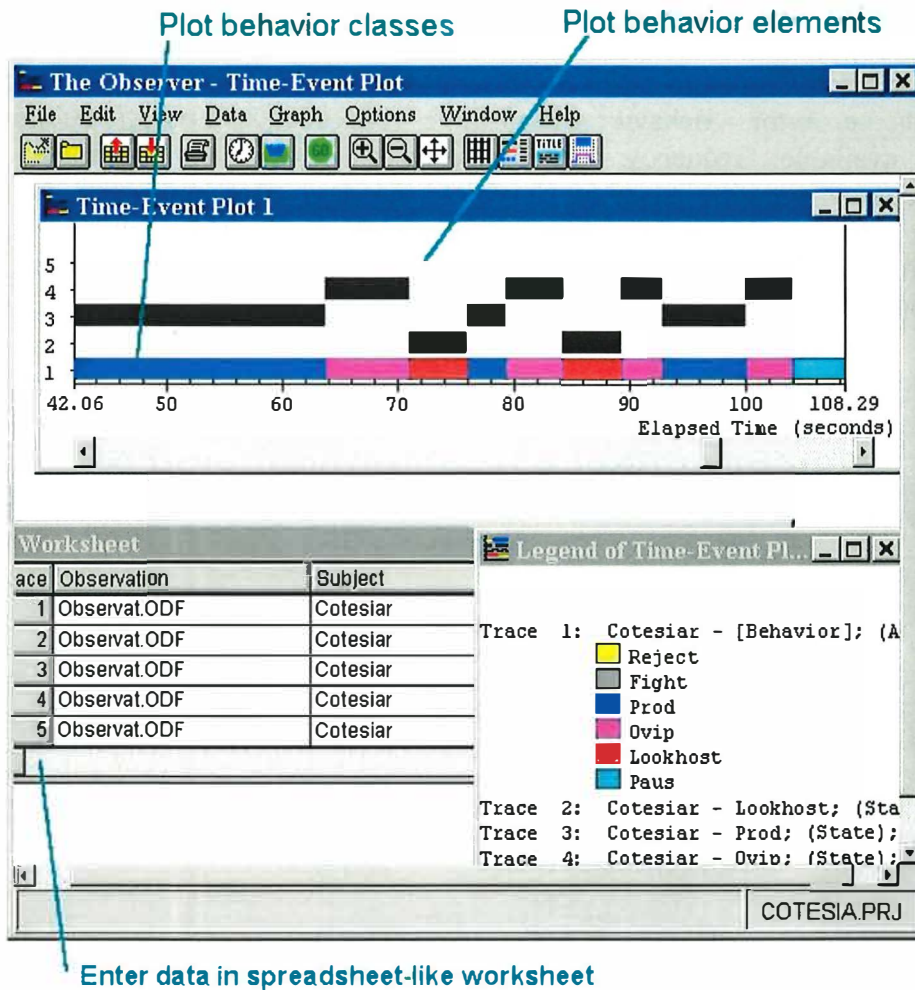


Figure 16: Example of Time-Event Plot

Critique:

The time-event plot actually allows the visualisation of the same data as the time-event table, the only difference being that this plot shows the duration of each action in an explicit way, i.e. through the length of bars. It is thus a tool providing a more immediate and visual aspect of the same data. Furthermore, the use of colours facilitates the reading of this data. Unfortunately, the possibilities of interaction are very poor: a zoom is available, but that's all. It is not possible to have details on demand, and it is thus not possible to get more information, for instance, by clicking on the graph. The same remark as the one made for the WebTrends stacked column chart applies here.

Elementary Statistics (Textual view)

The term Elementary Statistics refers to descriptive statistics that are calculated for specific events, i.e. Actor – Behavior – Modifier 1 – Modifier 2 strings. The following statistics are available: frequency, relative frequency, latency, total duration, total duration (%), mean duration, standard deviation, standard error, minimum duration, maximum duration. This analysis can be run for one observational data file as well as for several files simultaneously, and in this case, the statistics can be calculated per observation or across the observations.

The time window is available.

		Still	Move	Missing Mouse Click	Click	Drag	No drag
	Behavioral class	Mouse Movement	Mouse Move	Mouse Click	Mouse Click	Mouse Click	Mouse Click
Data1	Total number	1	1	0	0	0	1
Data1	Rate	0.45	0.45	0.00	0.00	0.00	0.45
Data1	Total duration	00:45.46	01:29.12				02:14.58
Data1	Total duration %	33.78	66.22				100.00
Data1	Mean duration	00:45.46	01:29.12				02:14.58
Data1	Median duration	00:45.46	01:29.12				02:14.58
Data2	Total number	3	1	3	1	0	3
Data2	Rate	0.10	0.03	0.10	0.03	0.00	0.10
Data2	Total duration	00:53.33	30:01.88	29:23.20	-		01:32.01
Data2	Total duration %	2.87	97.13	95.04	-		4.96
Data2	Mean duration	00:17.78	30:01.88	09:47.73	-		00:30.67
Data2	Median duration	00:16.74	30:01.88	00:13.01	-		00:29.72

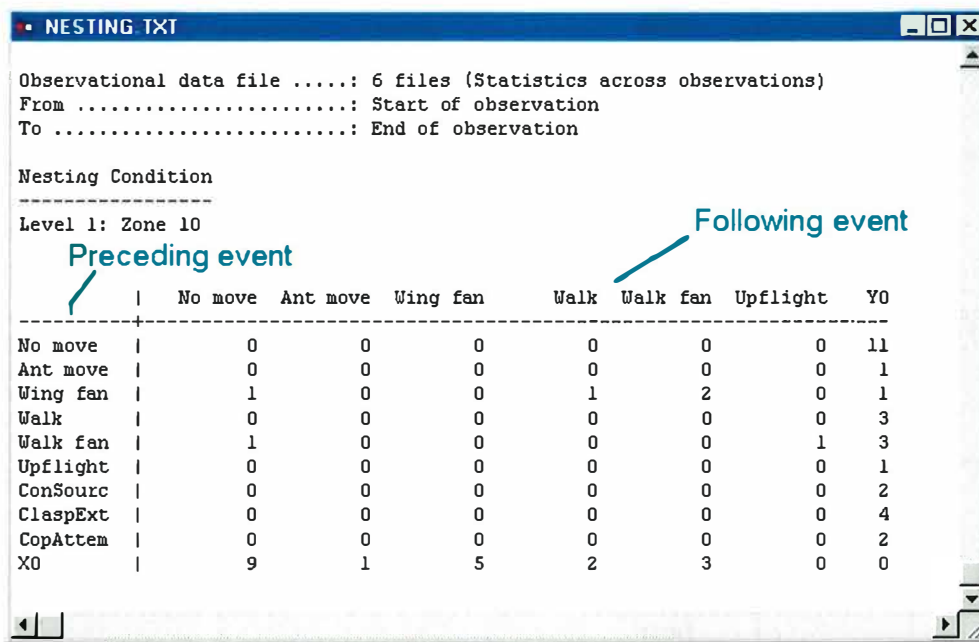
Figure 17: Example of elementary Statistics.

Critique:

Those statistics are presented under a textual form that seems to be quite appropriate for this kind of data. Indeed, much different information is displayed, and it would be impossible to represent them in a graphical way. Furthermore, this text format allows one to have a precision that would be difficult to keep in a graph. On a design point-of-view, it might have been useful to present those statistics under a grid format, which could have avoided some hesitations caused by the lack of lines and columns. Concerning the contents and the nature of the available statistics, each of them might be useful, depending on the context.

Lag Sequential Analysis (Textual view)

This technique examines how often certain events are preceded or followed by other events. The program calculates the frequency of transitions between pairs of events or states. The first event of each pair is referred to as the *criterion event*, while the second event is the *target event*. The lag is the interval separating the two. For instance, in a usability evaluation, The Observer tells how often a certain operation is followed by an error. It is possible to choose between state lags or time lags and set the lag length.



Observational data file: 6 files (Statistics across observations)
From: Start of observation
To: End of observation

Nesting Condition

Level 1: Zone 10

	No move	Ant move	Wing fan	Walk	Walk fan	Upflight	Y0
No move	0	0	0	0	0	0	11
Ant move	0	0	0	0	0	0	1
Wing fan	1	0	0	1	2	0	1
Walk	0	0	0	0	0	0	3
Walk fan	1	0	0	0	0	1	3
Upflight	0	0	0	0	0	0	1
ConSourc	0	0	0	0	0	0	2
ClaspExt	0	0	0	0	0	0	4
CopAttem	0	0	0	0	0	0	2
X0	9	1	5	2	3	0	0

Figure 18: Example of Lag Sequential Analysis

Critique:

This analysis is not easily readable, and we think that a graph format could have been easily considered. Indeed, this interesting data seems completely appropriate for a graphical display, which would probably be much more visual than this textual and very poor format. Of course this table is understandable, but it appears to us that a graphical format would allow us to visualise the data more easily and quickly, while being even more readable.

Conclusion

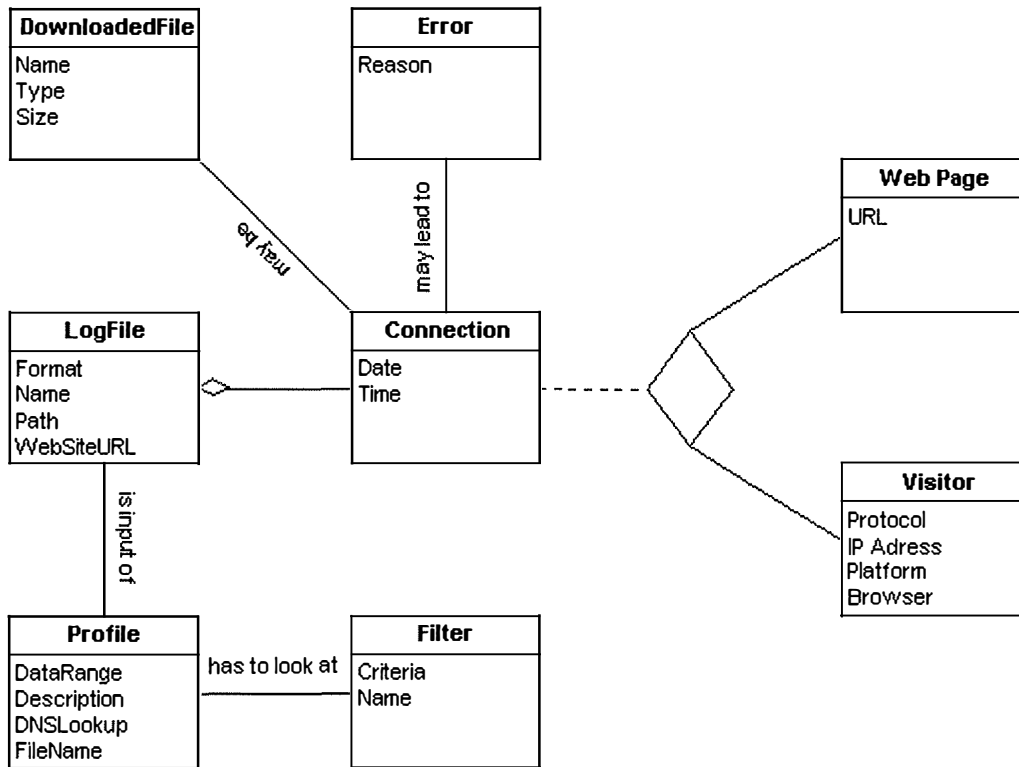
A global remark to be made is that only one chart can be visualized at the same time. Now, the fact is that sometimes, the analysis would be made easier and more interesting through comparing the same data displayed under different shapes. Furthermore, the same remarks as the ones formulated for the WebTrends charts can be applied here: there is nearly no possible interaction. Furthermore, a recurrent remark that can be applied to all those tools is the problem of the interactivity. It is generally impossible to easily filter data, and to get details on demand.

3.2.3 Data Analysis

3.2.3.1 WebTrends

Extant data model (Class Diagram⁵)

We represent the data involved in the process of analysing log files through the following Class Diagram.



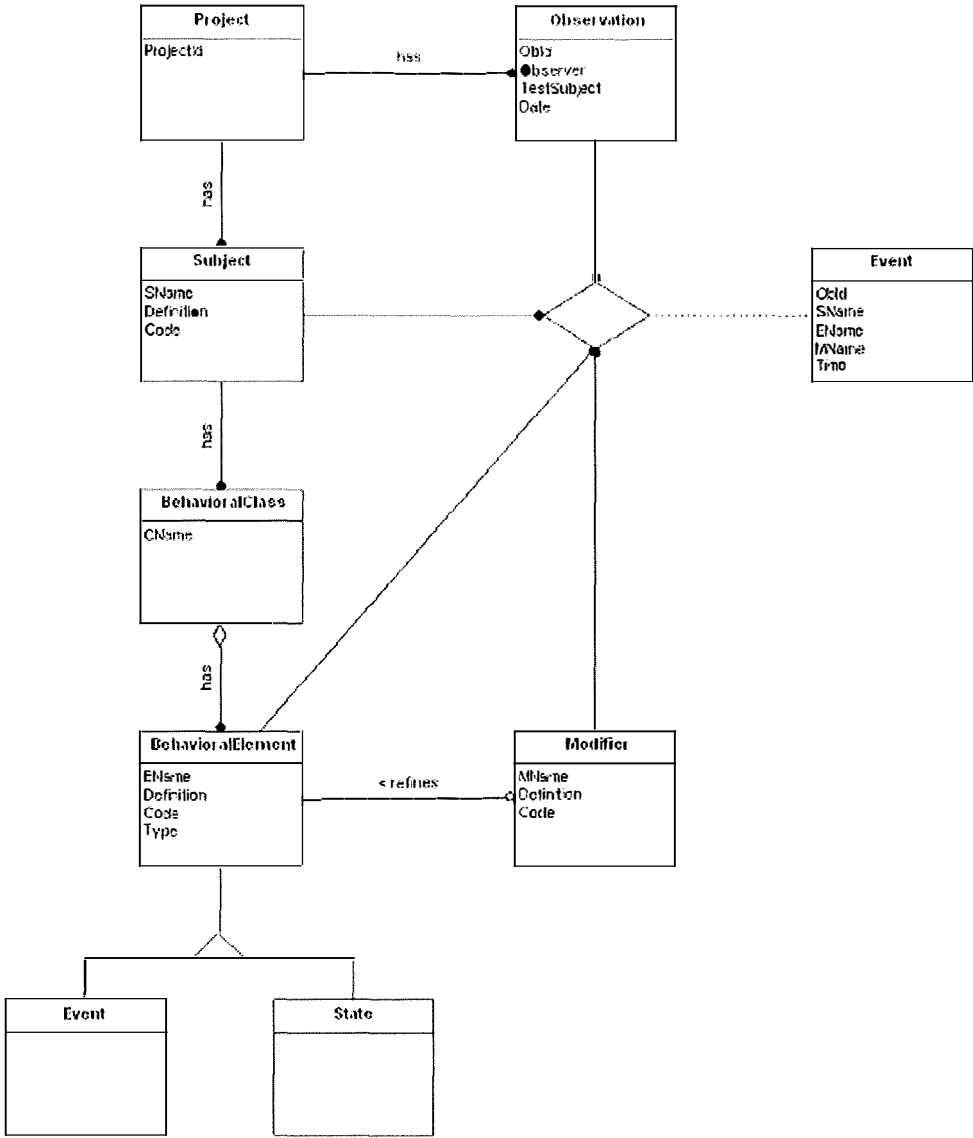
Comment

When a visitor is visiting a web site, some information about the actions carried out is stored in a log file. A log file is identified on the web side server by its format, name, path and the URL of the site it is related to. The log file is made up connections. A connection is defined by the date and the time of the connection and results from the association between a web page URL and a visitor, characterised by his protocol and IP address as well as the platform and browser used. A connection may be a file to download in which case, this file can be described by its name, type and size and may also end in an error. In order to report on a web site, the Log Analyser uses a profile. This profile is identified by its description and has as input, the log file to analyse, the type of data (data range) to look for, the File name for the output report and a number of filters that select, according to some criteria, the information that need to figure in this report.

⁵ For a short summary of the Class Diagram notation, see Appendix B.

3.2.3.2 Noldus

Extant data model (OMT Notation)



Comment

This diagram describes the links existing between the different types of data used in the Noldus Observer Video Pro, and allows the understanding of the way the concepts explained above are hinging with each other.

At first, it is obvious that the Project class is a kind of “Root” class from which the rest of the graph is spreading, a project having a certain number of subjects. These subjects are thus defined for a project, and so belong to this project’s configuration. In a slightly different way, one project has a certain number of observations (0 or more), each of them realised in the framework of this project. Each observation contains the following data: its id, the observer’s id, the name of the tested subject, and the date.

Each subject, as said in Appendix A, has got a name, a definition and a code, the definition being a completely descriptive field.

0 or more behavioral classes can be associated to one subject (through channels, see in the Task Analysis of Noldus), and each behavioral class is composed of a certain number of behaviors, which is shown through the aggregation symbol.

The two classes “Event” and “State”, linked to the behavioral element class through an inheritance relationship, are, as described in the Task Analysis of Noldus, two different types of behaviors, and are thus two subclasses of the behavioral element class.

Of course, as said above, modifiers are there to give more precision, and only two of them can be associated to one behavioral element.

There is still the “Event” association to consider: this association sets a relationship between an observation, a subject, a behavioral element, and a modifier. This class’s fields are: the observation id, the subject’s name, the behavioral element’s name, a modifier’s name, and the time.

So this diagram reflects the different parameters belonging to the configuration, and the link that can be made between them through an event.

3.2.4 Conclusion

3.2.4.1 WebTrends

The WebTrends Log Analyser appears to be very a good tool for analysing web sites' activities. Although it is limited to the analysis of log files, this software is well designed, involves simple procedures and provides understandable and detailed reports.

On a usability point of view, it appears thus to be pretty good software. For instance, when performing the different tasks described in section 3.2.1.2.1, one is never really lost since the help is constantly available and very clear, and one is generally well oriented by the program, i.e. there are not many possible choices and thus many possibilities of errors in the process of a task. Therefore, as we have already concluded above, we consider WebTrends as a success at the level of usability.

Furthermore, we want to underline the quality of the available charts which are in our opinion quite speaking and allow the visualisation of the data in a very interesting way.

However, the lack of interactivity we have pointed out in section 3.1.2.4.1 among those charts is the weakness of the system. It is a pity to see, for instance, that it might happen that one would not have the possibility to view all the data displayed in one chart, only because the chart cannot be moved or turned. We believe that more interactivity in the WebTrends visualisation tools would have made that program much better, and we will particularly focus on that aspect when designing our system.

3.2.4.2 Noldus

As a tool designed for the collection, analysis, presentation and management of observational data, the Observer has got quite a wide range of possible applications. It can be used to record activities, postures, movements, positions, facial expressions, social interactions or any other aspect of human or animal behavior, and in any context. It is thus not specialised in the field of usability testing, and from our point of view this is a real problem. Indeed, because of its very general scope, this program is quite limited and cannot afford to explore any issue more deeply and accurately. This is a handicap as far as subjects sometimes need to be handled with more accuracy than Noldus options allow. For instance, the data formats could be more appropriated.

Furthermore, the available visualisation tools are pretty poor and not adapted to our context. Much more adapted graphs, without being much more complicated, might be easily developed. For instance, the lag-sequential analysis contains interesting data, which could help the analysers to discover usability problems during the analysis. Unfortunately, the text format is difficult to read. A graphical format might be more speaking.

3.3. Investigation into the semantics of typical usability data

3.3.1 Introduction

The data coming from usability evaluation, also called usability data, is the object of this section. We first discuss the different kinds of usability data that may result from the different usability evaluation methods exposed in Chapter 2. From there, we will try to determine what is typical usability data, particularly in the context of data used by WebTrends Log Analyser and Noldus Observer Video Pro, in order to point out the most useful data to be visualised in our system.

3.3.2 Qualitative and Quantitative Data

Different kinds of usability data can be collected from the usability evaluation methods exposed in Chapter 2. A major distinction is to be made between two very different kinds of data resulting from usability testing and inspection methods: quantitative and qualitative data.

Quantitative data

Quantitative data can be defined as data which can only take specific numerical values. Such data is usually perceived as being more objective than qualitative data. Quantitative data is normally used in usability in two ways: first, it allows one to define quantifiable goals before the usability test is performed (e.g. if 70% of the users did not complete the task, then it can be considered as problematic), and second, it can be a direct descriptor of performance. Quantitative data is thus useful in situations where, for instance, a design decision has to be made. It is useful as well in providing summaries and overviews, since quantitative measurements can be generalised to the entire population with a known degree of accuracy.

Qualitative data

Qualitative data is extremely varied. Indeed it includes any information that can be collected and that is not numerical in nature. Such data is generally considered as being more subjective, less formal than qualitative data. Subjective data may include the user's comments, key words and key phrases, as well as their opinions and feelings about the object. The purpose of gathering such data generally is a deeper understanding of the human interaction experience. Qualitative data can be used for pattern finding, it can be used as an approximation of quantitative data when making a "first pass" at addressing an issue, and more important, qualitative data can be used to diagnose usability faults and prescribe solutions. Indeed, the use of data that elicits subject perspectives is especially important since observers do not always have the correct intuition about what is going on with the subject. It is useful in presenting personal information. Qualitative data can be collected from several usability methods, among which we can point out questionnaires, interviews, observation, focus groups, etc.

In order to achieve an evaluation that is as complete as possible, it is preferable to collect and analyse both kinds of data. Indeed, qualitative data is often described as being good for discovering problems, when quantitative data is good for discovering those problems' causes. Thus, while quantitative data may provide answers to questions such as "how much", "how many", or "how often", qualitative data can rather answer the question "why". Of course this is not an absolute rule, and both kinds of data can have different or similar advantages and disadvantages, depending on the situation.

This basic distinction allows us to cover all the different kinds of data resulting from any usability evaluation, and therefore, we have to investigate more deeply those two kinds of data in order to find out their common characteristics, and thus the characteristics determining and defining typical usability data.

3.3.3 Common Characteristics

Obviously, those kinds of data have very different natures, since quantitative data basically consists of numbers and statistics, objectively stored, that are used to describe the results of a usability evaluation performed by a user, and qualitative data is composed of subjective opinions, given by the users in questionnaires or interviews, that can take many shapes such as comments, impressions, etc (even though, sometimes qualitative data may be assimilated to quantitative data; it is the case for instance with rigorous questionnaires, based only on closed questions, where the users have to supply a simple fact, or to state their opinion on a rating scale).

It seems therefore logical to consider that both types of data have no common characteristic at the level of their composition, i.e. of the kinds of basic data constituting them, since we have an opposition between numerical data and non-numerical data. That is why, in order to find out the common characteristics that might tie both types, we have to consider another level, which is the one of the interpretations and discoveries the collected data should allow us to realise.

Of course, the overall goal of any usability evaluation method is the amelioration, if possible, of the tested program's usability, through discovering its deficiencies and solving them.

In the context of quantitative data, this is normally done through goal setting [Nielsen, 1993], which means that the different usability parameters one wants to assess can be expressed in measurable ways. Before testing, it is important to specify the goals of the user interface in terms of measured usability. For each usability attribute of interest, several different levels of performance can be specified as part of a goal-setting process. One would at least specify the minimum level which would be acceptable, for example, for a release of the product; but a more detailed goal specification may also include the planned level one is aiming as well as the current level of performance.

In the context of qualitative data, however, the situation is rather different: indeed each usability evaluation method leading to qualitative results has its own particular objectives, that are often slightly different from a method to the other. But basically, the common basis of all those methods is to get deeper insight into the discovered problems by knowing the users' opinions.

Again, we can conclude here that the goals underlying the collection of both kinds of data do not really have inclusive parts.

3.3.4 What is typical usability data

3.3.4.1 Different kinds of data

We can say that usability testing typically involves two kinds of data:

Data collected while listening to the users (i.e. qualitative data)

Logged data (i.e. quantitative data)

The data logged for each participant when interacting with an object generally includes:

- Facial expressions
- Overall body movements
- Interactions with the object

Again, if a computer is used within the session, then the images from the computer's screen may also need to be captured. When capturing a session that also involves the researcher then similarly, his or her interaction with the participant may need to be captured as well.

The above considerations lead us to the conclusion that both kinds of data do not have any common element, neither at the level of their nature, nor at the level of their objectives. Since this project will focus only on the data resulting from tests performed in a usability laboratory, i.e. logged data, we will focus only on quantitative data and we will restrict the scope of this analysis to an investigation into the semantics of typical quantitative usability data.

Common characteristics that should be present in any quantitative usability data can be identified:

- The collected data should allow us to perform the following measurements [ISO 9241, 1998]:
 - Effectiveness: it is the accuracy and completeness with which specified users can achieve specified goals in particular environments [van Greunen, 2002].
 - Completion rate:
Percentage of participants who completed each task correctly.
 - Mean goal achievement:
Mean extent to which each task was completely and correctly achieved, scored as a percentage.
 - Errors
 - Assistance (e.g. help screens, human interventions)
 - Efficiency: it is the resources expended in relation to the accuracy and completeness of goals achieved [van Greunen, 2002].
 - Task time:
Mean time taken to complete each task
 - Completion rate efficiency:
Mean completion rate/mean task time
 - Goal achievement efficiency:
Mean goal achievement/mean task time
 - No of references to the manual:
Number of separate references made to the manual
 - Productive period:
Portion of time the user did not have problems.
 - Learnability:
Comparison of the user's and expert user's efficiency for a task.

3.3.4.2 Conclusion

It is actually very difficult to establish clearly what is typical usability data, because the data involved in the process of evaluating an object may vary considerably from an analysis to another. For example, very detailed and accurate data may result from a good quantitative log analysis while the outcome of a qualitative analysis such as questionnaires and interviews is usually composed of very subjective opinions and comments expressed by the user. It is thus pretty difficult to give a general though accurate definition of typical usability data, and we therefore restrict the definition to typical quantitative usability data for which common characteristics are listed above.

3.3.5 Typical quantitative usability data: case studies

Let us be back with software analysed before, i.e. the WebTrends Log Analyser and the Noldus Observer Video Pro, and let us try to determine for each of them the typical data present in the extent data models (cf. Section 3.1.3) that is necessary to fulfil the requirements for typical quantitative usability data.

3.3.5.1 WebTrends

The data used by the log analyser is an example of web-oriented data. The log files used are a typical example of automated logged data. Among this data, as explained above, we generally make the difference between *Server-side logging* and *Client-side logging*.

In WebTrends' case, we face server-side logging which involves very different data from most data involved in the evaluation of objects. That's why there are no facial expressions, no body movement, no images of computer's screens and of course no interaction between the researcher and the web user. Though log files analysis is one of the best ways to collect usability data as it involves a lot of participants without any expensive and time-consuming tests, it also has the inconvenient that it is absolutely impossible to establish whether a user has performed the task he wanted to perform. Furthermore, interpreting the actions of an individual is extremely difficult because, for instance, web caches (both client browser caches and Intranet or ISP caches) can intercept requests for web pages, and so these requests might never reach the server and thus not be logged. Another problem is that several users might share the same IP address, making it difficult to distinguish who is requesting what pages.

To analyse a web-based usability test, a few important indicators can be identified. These include identifying the various paths users take, recognizing and classifying the differences in browsing behavior, knowing key entry and exit pages, and understanding various time-based metrics (e.g., average time spent on a page, time to download, etc.). All this data, given the framework of a task and the means to analyse it, can help designers discover where users encounter obstacles such as confusing navigation, difficult to find links, missing information, etc.

Now, we have to keep, according to the conclusion we have reached in the typical usability data analysis, the only elements that are necessary and sufficient in order to get the different measurements described above.

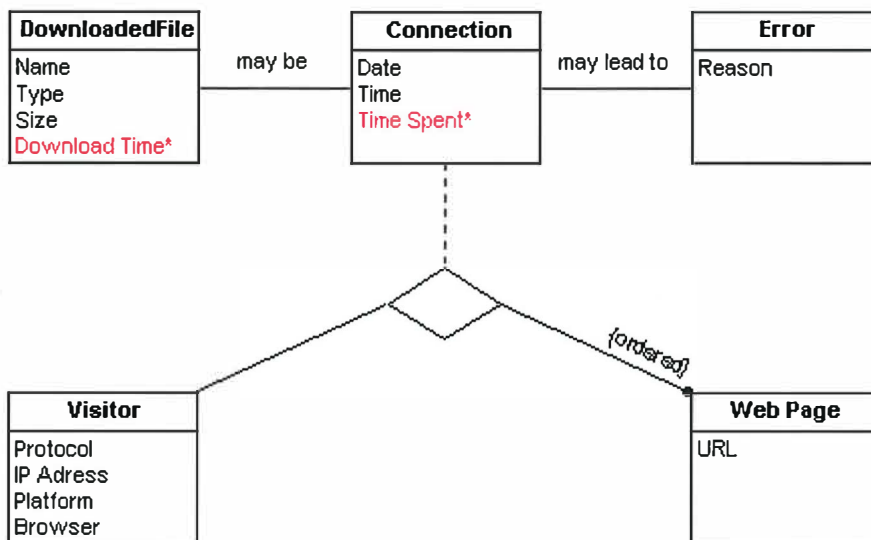
In order to compute the completion rate, it is necessary to know the number of participants, and to distinguish different visitors. This supposes the possibility to identify them, and thus to the need to keep the Visitor class.

We would also like to know if the user completed the task or not. Since no attribute or class specifically indicates that information, we have to know the sequence of the different actions performed during the task. Here three classes allow us to know that information: the Web Page class, the DownloadedFile class, and the Error class. We notice that the Error class contains as well the information allowing us to know the Errors measurement.

Since a Log File instance contains all Connexion instances, it is only a summary of useful usability data, but is no usability data in itself since it cannot be collected during a usability test. This class is actually somewhat too close to the functioning of WebTrends. And so are the Profile and Filter classes, which serve to the intern functioning of WebTrends too.

The time to download a file as well as the time spent on a connection have been added, since they are not available in WebTrends, but are necessary in order to compute the mean time taken to complete the task.

The classes responding to the above-mentioned requirements for a web-oriented data model are presented in the class diagram given below on Figure 19:



* Not available with WebTrends Log Analyser

Figure 19: WebTrends typical usability data diagram

3.3.5.2 Noldus

The Observer offers two different modes of data entry: coding, using pre-defined codes for actors, behaviors and modifiers, and annotation, i.e. entering free-format text strings. These methods can be used simultaneously. Their combination makes the Observer suitable for quantitative as well as qualitative behavioral studies.

However, the main and mostly used data produced by the Observer consists in events recorded during an observation (events can be scored while the video recording is being made, or it is possible to score an existing video recording). Each event is a combination of actor, behavior and modifier, and is logged with the time at which it occurs.

Again, we have to keep, according to the conclusion we have reached in the typical usability data analysis, the only elements that are necessary and sufficient in order to compute the different measurements described above.

We want to know about the completion rate, which supposes to be able to count the number of participants. We thus keep the Subject class, which allows us to identify the different subjects performing tasks, and to make the required information available.

We want to know whether the task was completed or not. This information is not directly available in the graph, but may be deduced from the class representing the behavioral elements. Indeed, by analysing their sequence, it is possible to know whether the participant reached the task's end or not.

The mean time taken to complete each task supposes to know all the task times. They can be partly obtained from the event time, but in order to get more accuracy with regards to this information's semantics, we keep the Event and State classes. However, not to stay too close to the Noldus terminology, and in order to avoid any confusion between the two classes called Event, we have decided to rename them. The Discrete class corresponds to the Event class, and the Continuous class to the State class.

At last, we keep the Event association. Even if the information it contains might be redundant with information contained in other classes, it constitutes the diagram central element since it regroups the most useful and interesting data.

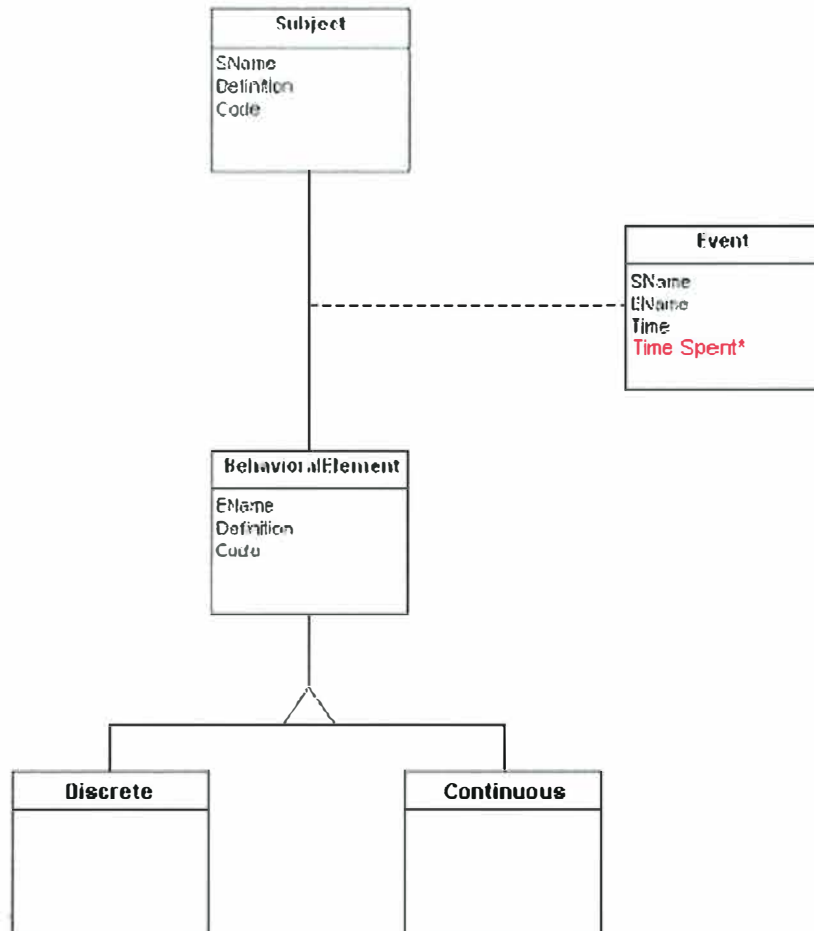
The Modifier class, which aims at refining the BehavioralElement class, is not necessary in the framework of the above-mentioned required data, since it only brings more detailed information about a behavior's description.

The Project class contains only data useful for Noldus, and thus presents no relevant data with regards to the quantitative data described above.

The observation class does not really bring interesting details. It does not allow one to compute any of the quantitative data mentioned above, and is removed from the graph.

As said above, the mean time taken is not fully available with the classes we have selected. Actually, we need to add a field to the Event class in order to know each event's duration.

The diagram resulting from this selection is presented below:



* Not available with Noldus Observer Video Pro

Figure 20: Noldus typical usability data diagram

3.3.5.3 Typical Usability Data: General Schema

In the last part of this section, we illustrate through another class diagram what quantitative typical usability data is according to the usability data coming out from both the observer and the analyser. To do that, we just keep common and essential information of the two class diagrams above, that allow us, of course, to realise the different measurements that characterise typical quantitative usability data.

For the analyser, only the visitor's ID (which is now assimilated to the subject's ID) and the software environment of this user are useful, the information remaining, i.e. access to web pages, files downloaded, errors, can be generalised and considered as events. The Connection class thus still has to be interpreted as representing a connection, but without forgetting that in the above WebTrends diagram, to one connection corresponds one of the above-mentioned generalised events, and that a connection can be seen more as a behavior than as a connection in the term's strict sense.

Concerning the observer, the Subject class stays, but not its definition and code attributes. The same remark can be applied to the BehavioralElement class, which (for a question of generality) becomes the Behavior class. The Discrete and Continuous classes, since they were specifying only whether the behavioral element discrete or continuous and did not contain any other interesting information, can be replaced by a field in the Behavior class: Type. The Event association is still the central component.

We have decided to add two subclasses to the Subject class: Device and User. This is to define precisely this class's semantics. The choice to consider User and Device can be explained from two perspectives: first, because of this work's context (usability tests do not make intervene any other actor), we restrict the Subject class's scope, and second, we add the Device class because we consider it brings very important information to the analysis.

If both the Connection and the Action classes are kept, it is because of their slightly different semantics which in our opinion deserves to appear in this diagram, even if the behavior class is sufficient to represent the data.

The diagram on Figure 21 shows what is typical usability data according to both WebTrends and Noldus. This graph gives an overview of the type of data we can expect from any usability test and this will therefore be considered in the following as the minimum required input data in order to derive interesting results in our system.

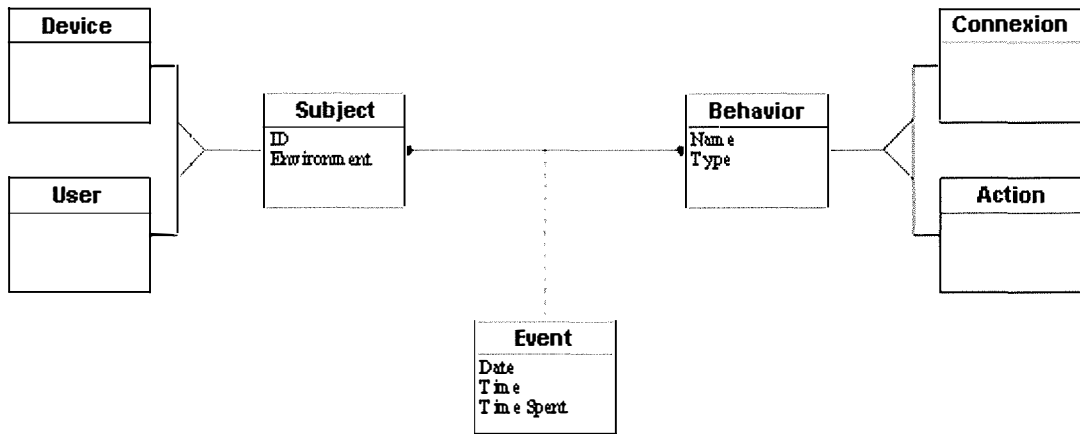


Figure 21: Typical quantitative usability data diagram

Chapter 4. An Analysis of the visualisation requirements

4.1 Introduction

The goal of this project is to develop a tool that can be used to interactively visualise usability data produced by any usability software and particularly by the Noldus Observer Video Pro. After some literature reviews, we have described some typical tools for usability analysis, i.e. the Noldus Observer Video Pro and the WebTrends log analyser, in order to point out the main usability problems present in these two extant systems. From there, we have defined what is typical usability data in general, and particularly with regard to the Observer and the Analyser. This analysis led to a list of typical quantitative measurements that should be visualised as well as a set of data that is necessary to allow this visualisation. The objective of this section is the statement of the major visualisation requirements for our future system.

We start by defining the methodology used to derive these requirements. Then we proceed to the statement of some limitations of the project's scope before going through the functional requirements, followed by the non-functional requirements.

4.2 Methodology

In order to derive the major requirements for this system, we base our arguments on three main sources.

First of all, we take into account the results of the extant system analysis developed in the previous chapter. We more particularly define our visualisation needs on the basis of the definition of typical usability data from the last section of that chapter.

The second source of information we use is the experience that some experts in usability testing shared with us during the five months spent at the university of Port Elizabeth. Among them, we can name Darelle van Greunen who is completing a Masters degree on the topic of formal usability testing.

Finally, we add to these sources our general understanding and intuition about the concept of usability acquired in the last ten months by some readings and works on the subject. The main bibliography for this literature can be found in the references mentioned in the three first chapters.

4.3 Limitations of the project's scope

4.3.1 The users of the system

Analysing usability data is an activity that involves mostly experts in the field of usability testing as well as designers of software. These can obviously be considered as expert users in computer science. That is why, in the following, we mainly focus our attention on the quality of the visualisation and on the interactivity of the graphs produced rather than on the simplicity of the system to which we attach less importance.

4.3.2 Type of data collected

As it is specified in section 3.2.4.1, we will focus, for the purposes of this project, on quantitative data and quantitative measurements. Furthermore, as our usability tool must be able to visualise data coming from usability software like the Observer, we will focus on logged data as described in the typical usability data section. This data involves observations of elementary actions or *behaviors* of a certain number of participants or *subjects* interacting with each other. We will optionally add to this data a video of the computer's screen if a computer is used within the usability test.

4.3.3 Subjects and behaviors

Although certain usability software such as the Observer can be used to record anything from activities, postures, movements, positions, facial expressions to social interactions or any other aspect of behaviors that human beings, animals or machines can have, we will limit the scope of this project to considering usability data generated from a usability test of computer software or other similar devices made up a screen and a number of ways to interact with the information displayed (e.g. a cell phone or a PDA). We will therefore limit the number of considered subjects to two: a human user and a computer (or another device). This limitation seems very important to us, since it is basically impossible to derive any useful visualisation techniques or statistics if we don't even know who or what is the object of the evaluation and what is the goal of this evaluation. We will thus consider observations of users trying to complete a certain task in a certain time, in any usability environment, in order to evaluate the usability of computer software or other device (theoretically the same for all observations). However, we will not impose any predefined actions or behaviors for the subjects as these can be anything depending on the tested device or the kind of behaviors the observer wants to take into account. Furthermore, we will not consider any classes or modifiers for behaviors and we will thus consider each of them separately.

4.3.4 Data sources

Though we are aware of the importance of an access to a multitude of data sources in order to allow the easy use of our system by a wide panel of users, we will limit the scope of this project to the design of a visualisation system for usability data without considering any interface with others systems. Of course, it should, for example, provide a usable interface with the Noldus Video Pro software, but designing a complete unit to access virtually any outside sources is beyond the scope of this thesis. We will thus refer to further works on the subject to deal with this problem.

4.4 Functional requirements

4.4.1 The object of the visualisation

Effectiveness and efficiency

Among the quantitative measurements developed in the definition of typical usability data (cf. section 3.2.4.1), the most interesting ones to consider and to visualise are the effectiveness and the efficiency.

As said in the above-mentioned section, the effectiveness can be evaluated through the following measures:

- Completion rate
Percentage of participants who completed the task correctly
- Mean goal achievement
Percentage of the content of the task that was correctly and completely achieved
- Errors
- Assistance (e.g. help screens, human assistance, manual)

However, although the percentage of the task the user completed correctly (or goal achievement) provide a more realistic impression of design quality than an absolute approach of success or failure that say basically nothing about why users failed or how well they performed the task they did not achieved, it is often impossible to measure it in a systematic way without having already analysed what the user has done. That is why, even if it might seem unreasonable to give the same score to both users who did nothing and those who successfully completed most of the task, we will only consider this percentage as an indication on the completion or not of the task. We can notice that the mean goal achievement will be in this case the same as the completion rate.

Furthermore, the number of errors is in our case very difficult, even impossible to determine. The main reason for this is that most errors consist simply of choosing a wrong path in the completion of the task. It is therefore impossible to evaluate, for example, the number of mistakes a user did if this user did not succeed to find the right way to complete the task. Other errors are sometimes automatically taken into account by the tested software, and can then be displayed in an error message. But again, it is not really possible to consider these messages to count, for example, the number of committed errors, as far as one message can be the result of several mistakes done before of the message is displayed (e.g. wrong or erroneous values detected at the execution of a function) in which case this message does not reflect the effective number of errors. The only solution to this problem is the definition of some specific behaviors for the errors that could be observed. But as we have decided not to predefine behaviors, it is impossible for us to recognise them and thus to use them for any purpose. Consequently, we will let the user of the system define the appropriate behaviors for errors and simply provide a way to determine their number.

The same argument can be applied to the assistance measurement, which cannot be evaluated without predefining certain behaviors. As in the case of errors, we will thus let the user define the behaviors providing a way to evaluate their number.

The efficiency is defined by the following elements:

- Mean task time
Mean time taken to complete the task
- Completion rate efficiency
 $\text{Completion rate} / \text{Mean task time}$
- Goal achievement efficiency
 $\text{Mean goal achievement} / \text{Mean task time}$
- N° of references to the manual

Again, it is impossible to consider the number of references to the manual without defining any behaviors and we will therefore ignore this element.

As a conclusion, the following measurements will be taken into account in the visualisation of effectiveness and efficiency:

- The completion rate or percentage of participants who completed the task correctly
By summarising the percentage of testers performing a task successfully, it is for example possible to get a quick look at how easy a particular task was for the testers. The ideal value for percentage of testers who did successfully is of course 100, and if this percentage is relatively low, this means that it should be investigated why the testers did so poorly on this task. If this number is very low, it indicates that the product may need to be tweaked so that it is more self-evident to the tester how to complete the task.
- The mean task time or mean time taken to complete the task
The mean time can be used to see how the group of testers performed as a whole on the task.
- The completion rate efficiency or completion rate/mean task time
The quotient of completion rate and mean task time specifies the mean percentage of the task completed successfully for every unit of time (per second for instance).
- The number of times each behavior has been observed across observations
This number can inform the analyst about the fact that, for example, the user has used the help function a high number of times, which obviously means that there is one or more problems. Actually, when any behavior is observed more often than expected, it generally means that attention should be paid to that fact.

However, it is above only question of information on all observations taken together but it can also be useful to consider the following measures separately for each observation or in comparison with the others:

- If the participant has achieved the task or not
Though this metric is quite coarse, it is a very telling one, and very easy to collect. Indeed it might be quite revealing to compare different users' results, in terms of their characteristics, such as their experience, age, ...
- The task time or total time taken to complete the task
It might be interesting, for instance, to compare the highest and the lowest completion task time. Indeed, this statistic can be very revealing if there is a huge difference between them, and can mean that it might be worth investigating if the high and low completion times are radically different, and if the tester performing the task with the highest completion time did not have as much experience as the tester with the lowest completion time.

- The number of times each behavior has been observed across the observation
When knowing how many times the same behavior has occurred, it can be possible to deduce things like a lack of self-evidence, the fact that an option is quite hidden to the tester, ... And if this number is very different from one observation to another, it might be interesting to investigate.

Other measurements

After some readings and an analysis of the visualisation requirements of the future users, especially Darelle van Greunen, we can propose a list of other measures that could be easily collected but also very useful as well in detecting what exactly are the design problems of the software. For each observation taken separately or in comparison with the others, the system should be able to visualise:

- The sequence of actions (user and/or computer behaviours) and the time spent for each action in the completion of the task.
Thanks to this information, it can be possible to follow the path each individual subject followed in performing the task and the time spent for example on each screen or subtask. Obviously, this measurement allows one to see, among other things, differences and similarities in the sequence of actions followed by different (types of) users, as well as detecting anomalies or divergences in the duration of certain actions. However, one has to be very careful with the interpretation of this comparison as a task can often be performed in several different ways.
- A comparison between the sequence of actions and an optimal sequence of actions or sequence of reference provided by the user of the system.
As described in [Nielsen and Mack, 1994], a description of the sequence of actions (i.e. behaviors) that the user should accomplish in order to complete the task in the most appropriate way given the current definition of the interface. These actions may be simple movements, such as “press the RETURN key”, or “move cursor to File menu”. Or they may be several simple actions that the user could execute as a block such as “login to the system” or “select save from file menu”. The decision as to what level of granularity is appropriate depends primarily on the level of expertise of the expected users. A guideline might be that reasonable collections of similar keystrokes, such as those used to input a filename, be considered as a single action [Wharton et al., 1992].

From there it can be possible to determine if the user followed an optimal navigation path by comparing his or her path with the optimal path. This statistic can be very speaking, as it can reveal very quickly where the tester went off the optimal and expected way, and thus where there might be a design problem. Through this information, it might be possible to detect more easily some problems such as backtrackings, turnings, ...

- The total time spent per behavior.
We can first notice that this information is only relevant for continuous behaviors as opposed to discrete behaviors. A discrete behavior is referred to as an instantaneous event, when a continuous behavior corresponds to a state, i.e. has duration. Discrete and continuous have basically the same signification than respectively “event” and “state” in the Noldus terminology. This statistic comes as a complement for other ones, and allows one to analyse the users’ general behaviors and trends. This information serves to determine where the user lost time in the completion of the task
- The number of times each behavior has been followed or preceded by another.
This can be interesting especially when considering behaviors such as errors or assistance. A help screen or a systematic unexpected action following another particular action might be an indication that the interface design needs some improvements.

Following the same idea as above, the last two points have also to be visualised for all observations taken together.

We can furthermore notice that all this information is even more useful when used in combination with a video of the device's screens.

Conclusion

As a conclusion for this section, here are the elements that need to be visualised:

For each observation separately or in comparison with others:

- If the participant has completed the task successfully or not
- The total time taken to complete the task
- The sequence of actions and the time spent for each action in the completion of the task
- The comparison between the sequence of actions followed by the user and an optimal sequence of actions
- The number of times each behavior has been observed in the completion of the task
- The total time spent per behavior in the completion of the task
- The number of times each behavior has been followed or preceded by another in the completion of the task
- The video of the observation if it is available

For all observations taken together:

- The completion rate or percentage of participants who completed the task correctly
- The mean task time or mean time taken to complete the task
- The completion rate efficiency or completion rate/mean task time
- The number of times each behavior has been observed across observations
- The total time spent per behavior across observations
- The number of times each behavior has been followed or preceded by another across observations

This exhaustive enumeration will be the starting point for the design of the visualisation metaphors in the following chapter.

Finally, we can add that all the elements listed above can be visualised and are relevant irrespectively of the device used (e.g. computer, PDA, cell phone, ...). However, certain characteristics of the device must be considered in order to interpret correctly the results of the usability analysis performed. A simple example of this is the importance of the device when performing a usability test relating to a connection to Internet. Surfing on the net is very different on the 21" screen of a computer or on the 1.5" screen of a cell phone. This explains probably, among other reasons such as the price, the relative failure of the WAP, and that is why, even if it is possible to perform the analysis in the same way whatever the device used, it is important to keep in mind some of its basic features. Given the great variety of existing devices and configurations, it is obviously impossible for us to take into account an exhaustive description of it. But what we can do is helping the user in one's task by giving to him some very general information selected on the basis of our intuition about this problem. These features are:

- The type of device (computer, cell phone, PDA, ...)
- The media available (text, image, audio, video)
- The screen size
- The number of colours
- The tools used to interact with the screen (keyboard, mouse, scanner, joystick, speech recognition ...)
- Other features to be defined by the user

4.4.2 The visualisation techniques

All we can say at the moment is that most of the information above should be visualised textually, graphically (if necessary using the 3D technology) and possibly using a video. Furthermore, the graphical views used should contain at least a main view and one or more additional views. The appropriate visualisation techniques used will be discussed in the following chapter about the design of the system.

4.4.3 Data access

The data should be accessed from a text file and should provide the necessary information to support the requirements defined in this chapter. The format of this file will be specified in the following chapter.

4.4.4 Data handling

According to Ben Shneiderman [Shneiderman, 1998], there are many visualisation guidelines, but the central principles might be summarised as this visual-information-seeking:

"Overview first, zoom and filter, then details on demand"

The 7 central principles are described below. For each of them we define the corresponding requirements for our system.

Overview Task

Principle

We can gain an overview of the entire collection to be visualised.

Application

It should be possible to visualise all the information given above simultaneously for all observations, all defined users and device behaviors, and for the whole period of time.

Zoom Task

Principle

We can zoom in on items of interest. Users typically have an interest in some portion of a collection, and they need tools to enable them to control the zoom focus and the zoom factor.

Application

Among the different available visualisation tools that will be discussed further (Chapter 5), some should provide the possibility to zoom in and out. It should be among others possible to zoom on the subjects, the behaviors and the time period. We can also notice that a big part of the zoom task can be performed by filtering the data we want to zoom on (c.f. Filter task).

Filter Task

Principle

When users control the contents of the display, they can quickly focus on the ones that interest them most by eliminating undesired items.

Application

The system should provide the user ways to filter data by selecting:

- The observations, directly or on the basis of the most useful criteria according to the experts in usability testing we met, i.e.:
 - The gender
 - The age
 - The language
 - The expertise (a user is either novice, medium, or expert)
- The subjects (user and/or device)
- The time period

Details-on-demand Task

Principle

One can select an item or group to get details.

Application

Details should be available on demand in some of the graphs that will be exposed in Chapter 5.

Relate Task

Principle

One can view relationships among items.

Application

The relations among items are provided by the elements to visualise, which are exposed in the "Object of the visualisation" section.

History Task

Principle

One can keep a history of actions to support undo, redo, and progressive refinements.

Application

The system should provide undo and redo functions.

Extract Task

Principle

One can allow extraction of sub-collections and of the query parameters in a file format that would facilitate other uses.

Application

The system should allow one to save the different views into a common file format and to print them.

4.4.5 Generalised task model

The functional requirements for the system described above can be summarised by the generalised task model below:

- 1. Open project containing observation files in a specified format**
- 2. See information about the observations**
 - 2.1. See general information about the observations (name, description, start date and time, total time, task completed or not)
 - 2.2. See information about the users (surname, name, gender, age, language, expertise)
 - 2.3. See information about the device (type, media available, screen size, colours, interaction tools, others)
- 3. Select specific observations**
 - 2.1 Select independent variables (i.e. gender, age, language, expertise)
 - 2.2 Select observation files
- 4. Filter data**
 - 4.1. Select subjects (i.e. device, user)
 - 4.2. Select behaviors
 - 4.3. Select time period

- 5. Visualise data**
 - 5.1. Visualise main graphical view
 - 5.2. Visualise additional graphical view(s)
 - 5.3. Visualise textual view
 - 5.4. View media file
- 6. Zoom data (in/out)**
 - 6.1. Zoom subjects
 - 6.2. Zoom behaviors
 - 6.3. Zoom time period
- 7. Undo/Redo actions performed**
- 8. View details on demand**
- 9. Save views to a file**
- 10. Print views**

As we will see in the following chapter, this predefined workflow benefits to be as dynamic as possible, allowing one to perform all of the actions mentioned above at every moment.

4.5 Non-functional requirements

The non-functional requirements of a system describe the level of performance that the system must meet in order to be accepted. This performance is largely concerned with the performance of the users' activities and translates the usability of the system [Newman, 1995].

These requirements are listed below:

4.5.1 Usability

The aim of such a system is of course to make the data easier to visualise. According to this goal, the easier to learn the system is, the easier to use the data will be. But, this system is designed for expert users, and the quality and performance of the representation is much more important than the usability.

4.5.2 Reliability

Visual representations must be as reliable and accurate as possible. In any case, the visualisation techniques might be less useful than raw usability data because of a lack of accuracy or because of techniques where very different data might have similar representations.

4.5.3 Speed of performance

The speed of performance is not a major concern for this system. However, the user must be able to visualise the data and to interact with it as quickly as possible. The chosen language as well as the hardware used will be the main factors for the speed of performance

4.5.4 Design constraints

The only design constraints that can be identified at this stage are the following:

- The system should be window-based
- The system should be platform independent

Chapter 5. Design of The System

5.1 Introduction

In Chapter 4, we described the functional and non-functional requirements of our system and the types of measurements that will be used.

In this chapter, we try to uncover the best ways to visualise this data by trying to find out the design requirements and criteria that apply best to it. We therefore focus on how to represent those elements in a usable and interactive way, that fulfils those requirements and criteria. In order to achieve this, we first go through the data requirements, in which we define with accuracy the necessary data for our project. This is done, among others, through the specifications of the file formats used by our system. Then we indicate the major design features and guidelines used for the definition the interface. First, the concept of multiple views is addressed and illustrated by the analysis of a typical multiple view system: Spotfire DesicionSite [Spotfire], then intervenes the design of each component, accompanied by screenshots of our program. To end, we tackle the question of the design of the visualisation metaphors; we begin with some general criteria that our charts should respect and we determine the charts formats which are best adapted to the data we want to visualise.

5.2 Data requirements

5.2.1 Data collected

In this section, we define the data we need in order to meet the visualisation requirements listed in the previous chapter. As a starting point for this analysis we go back to the list of elements to visualise in the conclusion of the section 4.2.1.

Here is the information about each information we need to satisfy these requirements:

- For each action performed within this observation:
 - The behavioral name of this action
 - The time elapsed in seconds since the start of the observation
- If the task has been completed or not
- An optimal sequence of actions to complete the required task
- A video of the observation

We can check the validity of this statement by evaluating if this data is enough for the purpose of our system.

For each observation separately or in comparison with others, we should be able to determine:

- If the participant has completed the task successfully or not
This information is directly available in the data collected
- The total time taken to complete the task
This time is obviously the time of the last action that has been performed within the observation
- The sequence of actions and the time spent for each action in the completion of the task
This sequence can be obtained by sorting all the actions collected in order of time, while the time spent per action can be easily calculated by subtracting the time related to this action to the time related to the following action in the sequence.
- The comparison between the sequence of actions followed by the user and an optimal sequence of actions
As the optimal sequence of actions is directly available, it is easy to compare it with the sequence resulting from the sort.
- The number of times each behavior has been observed in the completion of the task
This can be calculated by adding the different occurrences of each behavior.
- The total time spent per behavior in the completion of the task
This is simply the addition of the time spent for each different action
- The number of times each behavior has been followed or preceded by another one in the completion of the task
This information result almost directly from the sequence of actions observed
- The video of the observation if it is available
The video is directly available

Some more data must be available to provide information about the observations, the users and the device and to allow the selection of the observations and the filtering of the behaviors as it is required by the generalised task model.

The needed data to perform these tasks are:

- A description of the observation including the following information:
 - The name of the observation
 - A textual description
 - The start date
 - The start time
 - The total time (already available)
 - If the task has been completed or not (already available)

- A description of the user involved in the observation including the following information:
 - Surname
 - Name
 - Gender
 - Age
 - Expertise
 - Language

- A description of the device used including:
 - The type of device (computer, cell phone, PDA, ...)
 - The media available (text, image, audio, video)
 - The screen size
 - The number of colours
 - The tools used to interact with the screen (keyboard, mouse, scanner, joystick, speech recognition ...)
 - Other features defined by the user

- A description of all possible behaviors including for each of them its name, description, type (i.e. discrete or continuous) and the subject that performs this behavior

5.2.2 Data model

Class Diagram

From the data needed to meet the requirements of the system, it is possible to represent the data involved in the process of analysing usability data with our system through the following Class Diagram:

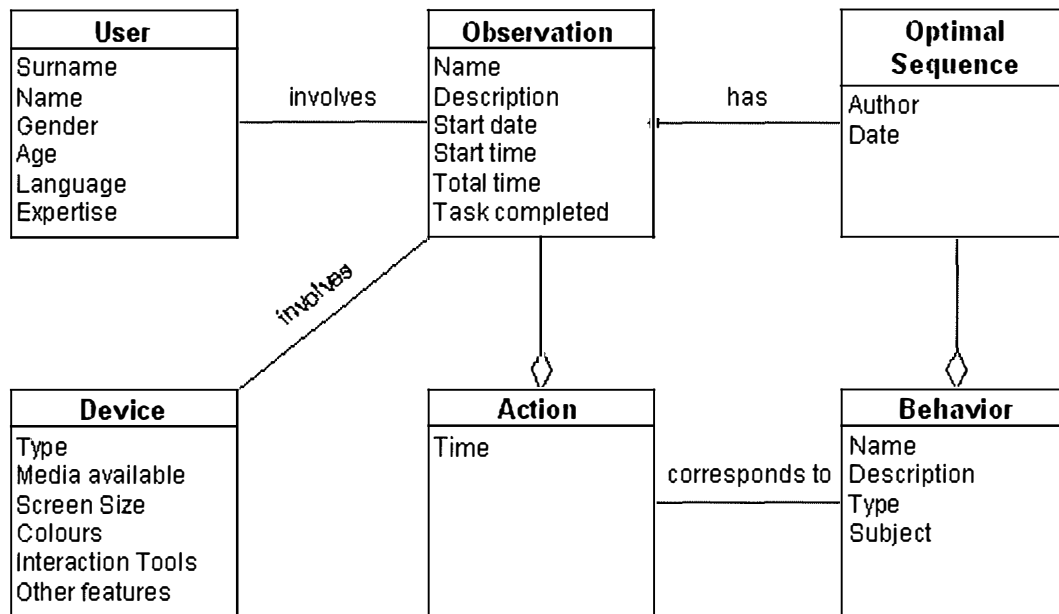


Figure 22: Data model

Description

To a predefined task corresponds an optimal sequence of behaviors to complete the task. A behavior is defined by its name, description, type (i.e. discrete or continuous) and the subject (i.e. User or Device) that performed this behavior. This optimal sequence goes for a set of observations. An observation is defined by its name, description, start date, start time, total time and an indication on whether the task has been completed or not. This observation involves a user (which is described by its surname, name, gender, age, expertise and language) and a device that interacts with that user. Furthermore, an observation is composed of a set of actions, each of them being identified by the time it has been performed and which corresponds to a particular behavior.

5.2.3 File format for the data

In this section, we define the format of the file containing the collected data. This data is split into two distinct files: a configuration file made up information common to all observations and an observation file for each single observation.

The Configuration File

This file must be in a text format and will contain the information common to all observations, i.e.:

- A list and a textual description of all possible behaviors including its type
- The optimal sequence of behaviors allowing the completion of the task (if available)
- The location of the media file related to this sequence (optional)

It must contain as well a list of all observation files related to this project.

The extension for the configuration file, i.e. which respects the following notation, is “div”, for Data Interactive Visualisation.

Description

We give here the main information helping the comprehension of this file format, but we insist that this description is under no circumstances an exhaustive one. For an accurate definition of the elements below, we refer to the BNF syntax in Appendix C.

In the following, the *italic* format refers to the required words of our format, i.e. the words that have to figure in it.

- At the top of the file comes the list of behaviors:

LIST OF BEHAVIORS:

Each behavior is defined as follows:

Name:

Description:

Type:

The only point here is the accuracy of the type definition, which must be either “discrete” or “continuous” and the size of the name which is limited to 10 characters. The behaviors associated to the user must come before those associated to the device. This is represented by the fact that the words *USER BEHAVIORS* precede all the user-related behaviors. These are followed by the words *DEVICE BEHAVIORS* meaning that the device-related behaviors follow.

- After this comes the optimal sequence of behaviors to complete the task (optional):

OPTIMAL SEQUENCE OF BEHAVIORS:

Each behavior is defined by its name in the list of behaviors.

- Then, if available comes the media attribute related to the optimal sequence, which must appear as follows:

MEDIA:

The Media indicates the location of the media file related to the optimal sequence of behaviors, and may begin with “.”. The presence of this point indicates whether the following path is a relative path or an absolute one. A point refers to a relative path, which means that the media file’s path will be considered from the directory in which the configuration file is. Then comes “\”, followed by as many folder names as desired, each of them being followed by “\”, and in the end is the file name, which must have as an extension either “mpg” or “avi” (This list can of course be very easily updated).

- Finally, we have the list of observation files:

LIST OF OBSERVATIONS:

Each observation is simply described by the location of the corresponding file on the hard drive. The same conventions as for the media file location can be applied. An observation file must have a “dof” extension (for Data Observation File).

Example

Here is an example of a configuration file that respects the syntax defined above:

LIST OF BEHAVIORS:

USER BEHAVIORS:

Name: Read

Description: The user is reading the screen

Type: continuous

Name: Key

Description: The user has pressed a key

Type: discrete

Name: Type

Description: The user is typing

Type: continuous

Name: LClick

Description: The user has left clicked

Type: discrete

Name: RClick

Description: The user has right clicked

Type: discrete

Name: MouseDrag

Description: The user is dragging the mouse

Type: continuous

Name: Inactive

Description: The user is inactive

Type: continuous

DEVICE BEHAVIORS:

Name: Screen1

Description: The screen 1 is displayed

Type: continuous

Name: Screen2

Description: The screen 2 is displayed

Type: continuous

Name: Screen3
Description: The screen 3 is displayed
Type: continuous

Name: HelpScreen
Description: A help screen is displayed
Type: continuous

Name: Error
Description: An error occurred
Type: discrete

OPTIMAL SEQUENCE OF BEHAVIORS:

Screen1
Read
LClick
Screen2
Read
Type
Key
Read
MouseDown
Key
Screen3
Read
Click

LIST OF OBSERVATIONS:

.\usability\observations\outlook\observation1.dof
.\usability\observations\outlook\observation2.dof
.\usability\observations\observation3.dof

The Observation File

The observation file must be in a text format and will contain the following information:

- The description of the observation
- The path of the media file
- The description of the user
- The description of the device used
- The sequence of actions performed by both the user and the device sorted on the time

Description

Like for the configuration file, we refer to the BNF syntax in appendix C for an accurate definition of the elements below.

- At the top of the file, comes the observation description attributes, which must appear in the following order:

OBSERVATION NAME:

OBSERVATION DESCRIPTION:

START DATE:

START TIME:

TOTAL TIME:

TASK COMPLETED:

- The Observation Name is limited to a maximum of 20 characters. All Observation Names should be unique. If they are not, a number will be automatically added to them to identify them.
- The Observation Description contains a textual description of the observation.
- The Start Date is composed of 4 digits for the year, immediately followed by “-”, followed by 2 digits, then “-“ again, and at last 2 other digit.
- The Start Time is defined by 2 digits, immediately followed by “:”, followed by 2 digits, then “:“ followed by two digits again, then “.” and at last 2 other digits.
- The Total Time indicates the duration of the observation in second. It starts with a number of digits, followed by “.”, and finally 2 other digits.
- The Task Completed field indicates simply if the task has been completed or not and takes the value "yes" or "no".

- Then comes the location of the media file for the observation (if available). This attribute is the same as the media attribute defined in the configuration file.
- Then we have the list of subjects, which contains the two subjects:

SUBJECTS:

Each subject has its own notation:

- The user:

USER DESCRIPTION:

Surname:

Name:

Genre:

Language:

Expertise:

- The device:

DEVICE DESCRIPTION:

Type:

Media Available:

Screen size:

Colours:

Interaction tools:

Other features:

We can first notice that the user always precedes the device. The user fields are quite simple. We simply specify the exact way to indicate the gender, i.e. either “male” or “female”, and the expertise, “novice”, “medium” or “expert”. The device fields indicate respectively:

- The type of device (computer, cell phone, PDA, ...)
- The media available (text, image, audio, video)
- The screen size
- The number of colours
- The tools used to interact with the screen (keyboard, mouse, scanner, joystick, speech recognition ...)
- Other features defined by the user

No particular constraints are set on these attributes.

- Finally we have the observation itself between the two following flags:

OBSERVATION START

OBSERVATION END

Between those is the observation list, which consists of a sequence of actions, each of them being characterised by the time elapsed since the beginning of the observation and a behavior name. The interesting points to underline here are:

- The observation must start at time 0.00,
- A behavior name in the list must correspond to a name of behavior in the list of behaviors of the Configuration File.
- The list is sorted on the time.

Example

Here is an example of file that respects the syntax defined above:

OBSERVATION NAME: Microsoft Outlook
OBSERVATION DESCRIPTION: this observation focuses on a specific component of Microsoft Outlook, the scheduling appointment function.
START DATE: 2001-11-16
START TIME: 10:30:00.00
TOTAL TIME: 97.28
TASK COMPLETED: yes

MEDIA: .\usability\observations\outlook\outlook.mpg

SUBJECTS:

USER DESCRIPTION:

Surname: van Greunen
Name: Darelle
Gender: female
Age: 16
Language: english
Expertise: expert

DEVICE DESCRIPTION:

Type: Computer

Media Available: Text, Image, audio, video

Screen Size: 21'

Colours: 16 bits

Interaction Tools: Keyboard, Mouse

Other features: Intel Pentium 4 1200 mghz, HD 40Gb, RAM 512 Mb, 512Kb cache, Microsoft Windows XP.

OBSERVATION START

0.00 Screen1

0.00 Read

5.81 LClick

5.83 Screen2

5.83 Read

15.71 Key

18.45 LClick

18.87 Read

24.18 MouseDrag

29.66 RClick

29.87 Type

38.98 LClick

39.00 Screen3

39.75 Read

42.03 LClick

42.04 Error

42.05 LClick

42.10 Inactive

46.36 Key

47.65 LClick

55.87 HelpScreen

56.12 Read

89.14 Key

95.20 LClick

OBSERVATION END

5.3 The design of the interface

5.3.1 Introduction

We try, in this section, to deal with the main design aspects of our system. We first introduce multiple views, before deriving some useful recommendations, partly inspired from the design of a typical multiple view application: Spotfire. Then, we define each of the components of our interface separately and with regard to both the functional requirements defined in Chapter 4 and the recommendations coming from the Spotfire analysis.

5.3.2 Multiple views

Definitions

Baldonado, Woodruff and Kuchinsky define in their paper "Guidelines for Using Multiple Views in Information Visualisation" [Baldonado et al., 2000] a *single view* of a conceptual entity as a set of data plus a specification of how to display that data visually. Note that a display may be either textual, or graphical. Views are *distinct* if they allow the user to learn about different aspects of the conceptual entity, e.g., by presenting different information, or by emphasising different aspects of the same information. They finally define a *multiple view system* as a system that uses two or more such distinct views to support the investigation of a given conceptual entity.

When to use multiple views

Deciding when and how to apply multiple views to information visualisation problems involves balancing a complex set of design tradeoffs. On the one hand, multiple views can provide utility in terms of minimising some of the cognitive overhead engendered by a single, complex view of data. On the other hand, multiple views can decrease utility when added to a system, both in terms of higher cognitive overhead (e.g., for context switching) and in terms of increased system requirements. As we present some guidelines, we identify how they significantly impact cognitive overhead and system requirements.

The cognitive aspects of an information management task include:

- The time and effort required to *learn* the system
- The *load* on the user's working memory
- The effort required for *comparison*
- The effort required for *context switching*

The impact on system requirements engendered by multiple views include:

- *Computational requirements* for rendering the additional display elements
- *Display space requirements* for the additional views

Wang Baldonado, Woodruff and Kuchinsky introduce four design rules to help designers assess whether or not multiple view systems are appropriate for their applications. These four rules are listed below:

Rule of Diversity

Use multiple views when there is a diversity of attributes, models, user profiles, levels of abstraction, or genres. A single view that accommodates many needs is likely to be a least-common-denominator view that is not optimal for any needs. Such a view may create significant cognitive overhead for the users by requiring them to simultaneously comprehend and assimilate a multitude of diverse data, some of which may not be relevant to their needs.

Rule of Complementarity

Use multiple views when different views bring out correlations and/or disparities. In a single view, a user may need to mentally extract and remember components they wish to compare. Maintaining and switching among these components can be cognitively demanding. Just as recognition is easier to perform than recall, so visual comparison is easier to accomplish than memory-based comparison. Multiple views leverage perceptual capabilities to improve understanding of relations among views.

Rule of Decomposition

Partition complex data into multiple views to create manageable chunks and to provide insight into the interaction among different dimensions. A single complex view can be cognitively overwhelming to a user. Multiple views can help the user to "divide and conquer," aiding memory by reducing the amount of data they need to consider at one time.

Rule of Parsimony

Use multiple views minimally. A single view provides a user with a stable context for analysis; multiple views incur the cost of context switching. Further, multiple views introduce additional system complexity. Accordingly, the designer must be able to justify the user's learning costs and the computational and display space costs of an additional view by appealing to the rules of diversity, complementarity, or decomposition.

Multiple views in our system

Given the four rules above and the number of elements to visualise (see section 4.2.1 for details), it is easy for us to justify the use of multiple views by our application.

First, we clearly meet the rule of diversity. As the elements to visualise cover a wide variety of information, it is absolutely necessary to split all that information using the appropriate metaphors to visualise each element or set of elements separately.

The rule of decomposition is in our case very similar to the rule of diversity. Partitioning the information into different views contributes to a better understanding and provides manageable data quantity inside views. The definition of the appropriate subsets of information to visualise simultaneously will be discussed in section 5.4.

The rule of complementarity is present as well. Even if it is early to talk about the content of a particular view, we can already say that visualised elements act as a complement for others in the investigation into the design problems of the tested interface. For example, an anomaly in the sequence of behaviors performed in the completion of a task can be easily investigated by evaluating the time spent on each behavior during that particular period of time, if necessary using the video.

Finally, the last rule (rule of parsimony) simply recommends using multiple views only when there is a compelling reason to do so. As additional views demand increased cognitive attention and take up valuable screen space, it is very important to consider the cost/benefit impact of introducing additional views. The appropriate number of views will be discussed in the section 5.4 about the design of visualisation metaphors.

How to design multiple view systems

The main goal of this section is to give the major guidelines for the design of multiple view systems. These guidelines are widely inspired by the Baldonado, Woodruff and Kuchinsky's paper about multiple views in Information Visualisation [Baldonado et al., 2000]. We will refer to these rules as many times as possible when designing the interface and the visualisation metaphors in the following sections. For further readings on the subject, we refer to North and Shneiderman paper: "Snap-Together Visualisation: A User Interface for coordinating Visualisations via Relational Schemata" [North, 2000].

The methodology for using multiple views can be partitioned into three main steps: Selection, Presentation and Interaction.

Selection

The first phase in the design process is the identification of a set of views. The selection task will be exposed in section 5.4.

Presentation

Once a set of views has been chosen, the designer faces a number of issues related to their presentation.

One of the first decisions a designer must make is whether to present multiple views side-by-side or sequentially. Even if the application allows the user to make this determination, a good default is still critical. To make this decision, the designer should consider how much space and time are available to the user, as well as how much space and time each of the candidate views requires. While it is relatively straightforward to compute space costs, time costs are often trickier to compute. Hidden time costs include the time required for a user to context-switch from one view to another and the time required for a view to be computed and rendered.

Another crucial point is to make relationships among multiple views more apparent to the user. Discerning the relationships among views can be a difficult task for the user of a multiple view system. The use of perceptual cues can make relationships more explicit to the end user. There are many types of perceptual cues that can be applied, but the spatial arrangement of views is the most commonly used cue.

Interaction

We next consider the interaction mechanisms supported by views. Each single view may have independent manipulation mechanisms. Often, these are tied together so that actions in one view have an effect in another view. One common interaction technique is *navigational slaving*, in which movements in one view are automatically propagated to other views. Another common interaction technique is *linking*, which connects data in one view to data in another view. A specific type of linking is *brushing*, in which the user highlights items in one view and the corresponding items in another view are highlighted by the system. Ideally, the interface for multiple views should be consistent and make the states of multiple views consistent. System state encompasses both the data set and the user's viewpoint. For example, if one view shows a particular region, a related view should show the same region. Similarly, if objects are highlighted in one view, the corresponding objects in a related view should be highlighted as well. For some applications, this consistency may not be desirable, e.g., because the user wishes to use different views to preserve different states. Also, as observed above, this consistency may not always be possible to implement. In these cases, that independence should be made clear.

5.3.3 An example of multiple view system: Spotfire DecisionSite

Spotfire DecisionSite overview

Spotfire DecisionSite has been especially designed for businesses to face today's challenges including the explosion in data and its migration to points outside the enterprise and to help solve these enterprise decision making problems by giving them the ability to effectively automate and manage delivered services and solutions for data access, dynamic visualisation and decision capture. An effective decision management should permit interactive exploration of the data and provide an environment where sequential problem solving, and "what if" scenario evaluation can occur. That is why Spotfire is based on ground-break research in dynamic queries, interactive visualisation and human-computer interaction and some of its features are therefore, as we will see, very interesting from our point of view. More documentation on Spotfire DecisionSite can be found on Spotfire site [Spotfire].

Methodology for the analysis

In order to have a quick look at the aspects provided by Spotfire that can be useful for the design of our application, we first examine the general principles that rule this system. Then we examine briefly how it illustrates the main functional requirements of our system summarised in the general task model of section 4.4.5. To do this, we try to see what can be done using a very simple file illustrating an example of the data that could be collected from a usability test involving a student trying to complete a predefined task on a computer. This file is similar to the one defined in section 5.3.2 about the file format for the data but in a tabular form and contains for each action that occurred within an observation:

- The number of the related observation
- The time in seconds elapsed since the beginning of the observation
- The behavioral name of this action
- The subject performing this action (Student or Computer)

Example: Observation 1 1.02 sec Read Subject

The complete file content can be found in appendix D.

Finally, we take the results of this analysis to derive some basic recommendations for the design of the interface exposed in section 5.3.3.

Spotfire general design principles

In this section, we give an overview of Spotfire main choices made for the three steps in the design of multiple views systems developed in section 5.3.2: the selection, the presentation and the interaction.

The selection

As we will see further, Spotfire provides a large variety of graphical and textual views to allow the visualisation of basically all kinds of data. The available views are therefore more exhaustive than designed and selected in a particular purpose.

The presentation

Spotfire design is first of all based on the dynamic, side-by-side display of a set of internal frames. These frames include graphical and textual views as well as filtering and informational views related to the items displayed. They can be quickly and easily added, removed, iconified and resized to provide a very effective way of investigation into the loaded data. However, the number of different views showing basically the same information makes relationships among these views not explicit at all and makes difficult the choice of the best view to visualise the data efficiently.

The interaction

Spotfire uses a *brushing* interaction, in which items highlighted in one view are highlighted in all displayed views.

Figure 23 shows an overview of all these principles together.

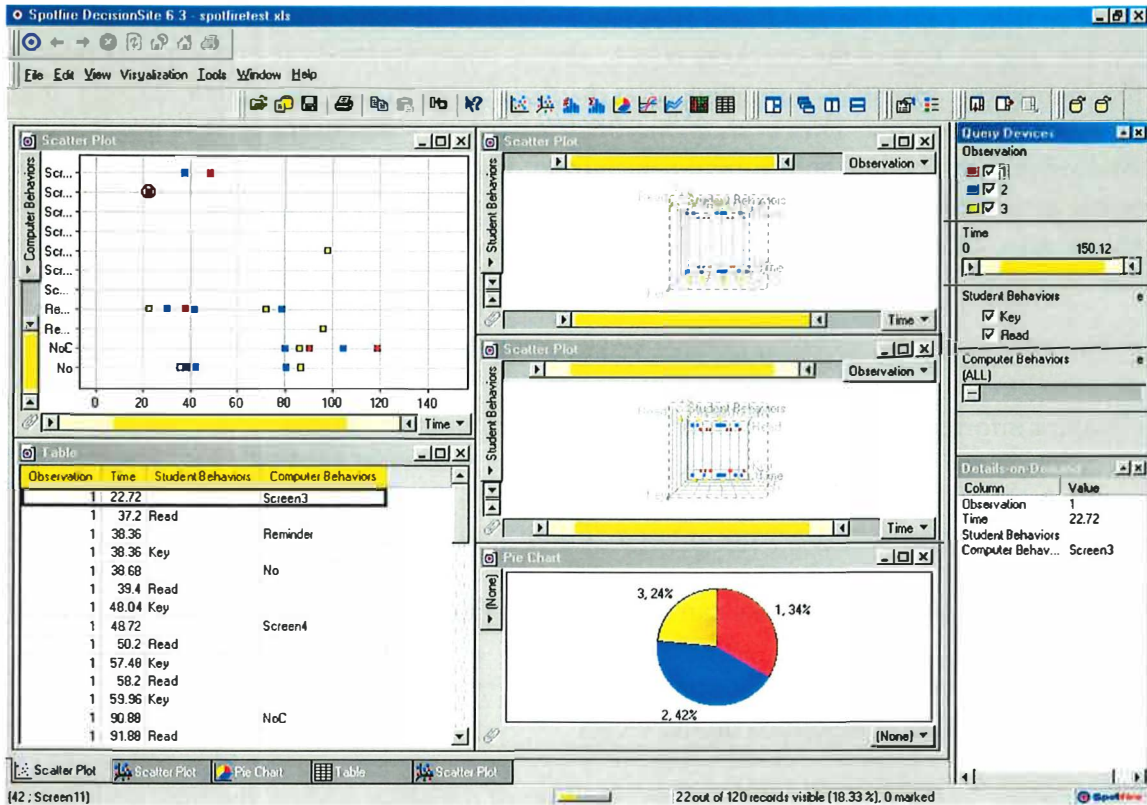


Figure 23: Spotfire general design principles

Illustration of the functional requirements of our system with Spotfire

1. The "Open project" Task

Spotfire enables integrated access to any kind of data sources, from local files and data warehouses to geographically dispersed databases.

It supports direct access to:

- Analysis files (Spotfire files)
- Text files
- Microsoft Excel files
- Data via the Microsoft windows clipboard
- Direct connections to local databases using the Microsoft ODBC or OLEDB technologies.

But also, through the server, to:

- Multiple relational databases
- Publicly available information on Internet

Spotfire files, Text files and Excel files can be opened directly using the Open option on the File menu. This is the quickest way to access unambiguously formatted data from a file. Similarly, properly formatted data on the clipboard can be pasted directly into Spotfire DecisionSite. However, if the data resides in a database, or if it needs some kind of pre-processing before being visualised, the "Import Data" function in the File menu is needed. This function is designed to help access the data, whatever the source and whatever the required pre-processing.

2. The "See information about the observations" task

No extra information about the observations is used in this analysis.

3. The "Select specific observations" and "Filter data" tasks

The selection and filtering tasks can be performed through:

- The "Query Devices" internal frame
- The selection functions on the views

The "Query Devices" frame allows the selection of the observations, the time period and the student and computer behaviors through checkboxes and sliders. What is more interesting is that it enables a constant feed of "what if" questions by automatically updating the information displayed. Through this, users can dynamically filter and query and get an immediate response in the system. To give a simple example of this feature, the selection of an observation in the "Query Devices" internal frame automatically and dynamically adds all the information related to this observation in the displayed views. More information about dynamic queries can be found in [Lanning et al. 2000].

The selection functions on the views allow one to choose what views are presenting. For example, the scatterplot shown on Figure 27 makes possible to select the information present on the X and Y-axis as well as the time period if these axis refer to the time.

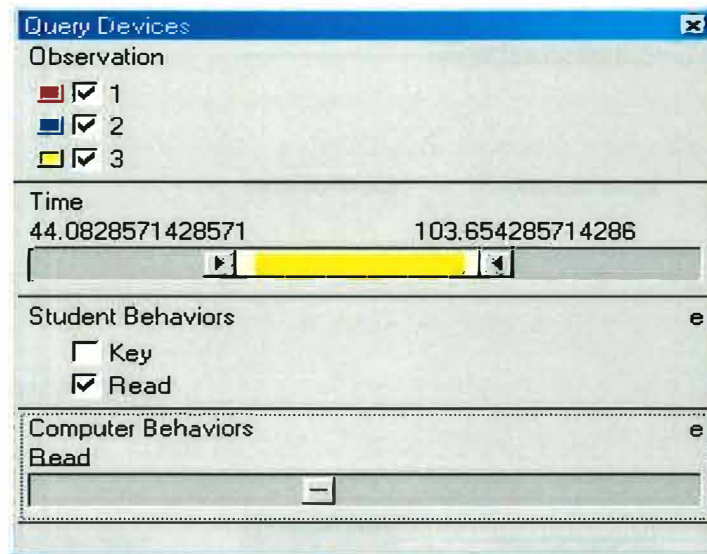


Figure 24: Spotfire "Query Devices" internal frame

4. The visualisation task

We simply give here five examples of views provided by Spotfire that approximately match our needs for the visualisation of usability data.

A Pie Chart is used to compare the total time of observations. Each colour on this chart refers to a colour in the "Query Devices" frame.

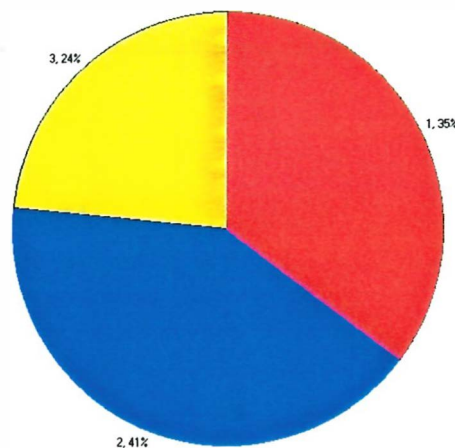


Figure 25: Spotfire Pie Chart

The histogram given below shows the number of times each selected behavior has been observed across selected observations.

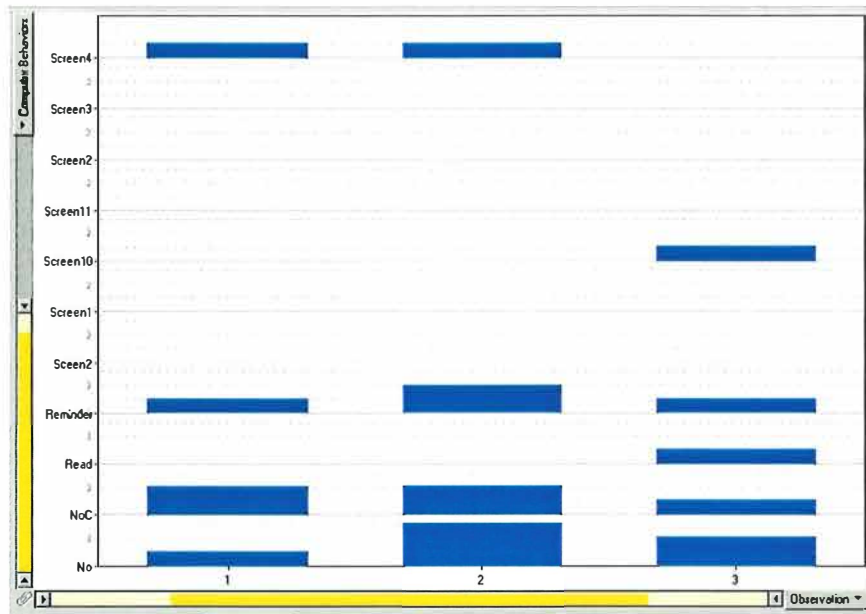


Figure 26: Spotfire Histogram

The Scatterplot on Figure 27 is useful to visualise what behaviors have been performed and the time they have been performed in the observations and time period selected.

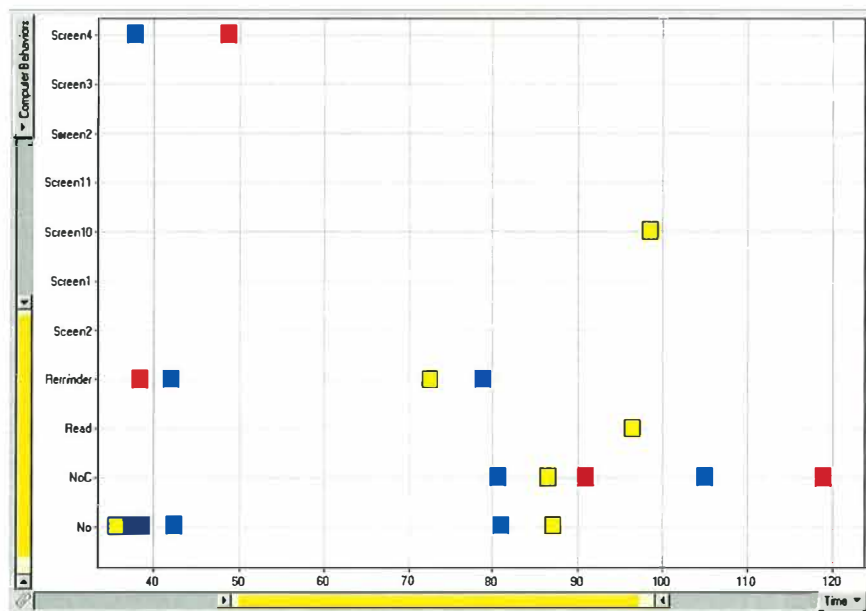


Figure 27: Spotfire Scatterplot

The Scatterplot 3D gives the same information as the Scatterplot except that behaviors that have been performed in different observations have different coordinates on a third axis (Z axis).

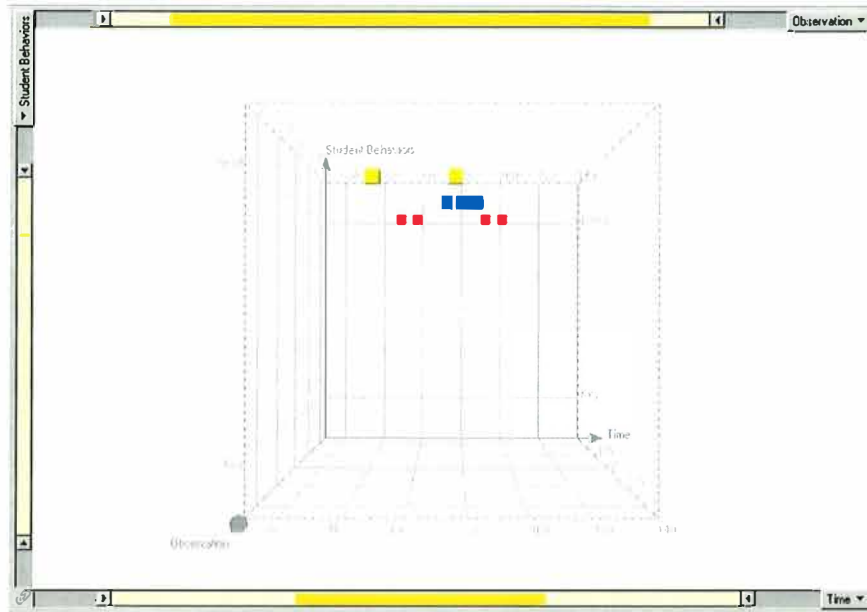


Figure 28: Spotfire Scatterplot 3D

Finally, the table below simply displays in a tabular form the information found in the file opened with regard to the current selection.

Observation	Time	Student Behaviors	Computer Behaviors
1	37.2	Read	
1	38.36		Reminder
1	38.36	Key	
1	38.68		No
1	39.4	Read	
1	48.04	Key	
1	48.72		ScreenA
1	50.2	Read	
1	57.48	Key	
1	58.2	Read	
1	59.96	Key	
2	37.56		No
2	37.84		ScreenA
2	38.38	Read	
2	42		Reminder
2	42.4		No
2	50.02	Key	
2	73.12	Read	
2	74.76	Key	
2	78.84		Reminder
2	79.28	Read	
2	79.8	Key	
2	80.68		NoC
2	81		No
2	82	Read	
3	35.68		No
3	42.11	Key	
3	47.96	Read	
3	57.26	Key	
3	72.54		Reminder
3	76.58	Read	
3	81.31	Key	

Figure 29: Spotfire Table

5. The Zoom task

The biggest part of the zoom task can be performed by selecting and filtering dynamically the items of interest. Furthermore, 3D views (Scatterplot 3D) provide the classical zoom functions dragging the mouse.

6. The "undo/redo" task

This function is not available in Spotfire.

7. The "View details on demand" task

This task can be performed by clicking on items of interest in the view displayed. An example of this is shown on Figure 30. Clicking on an item (the red square) of the Scatterplot displays all the necessary information about this item in the "Details-on-demand" internal frame.

8. The "Save View" task

An "Export" function is available in the File menu and allows the exportation of the views in the bitmap file format.

9. The "Print View" task

The print function is classically available in the File menu.

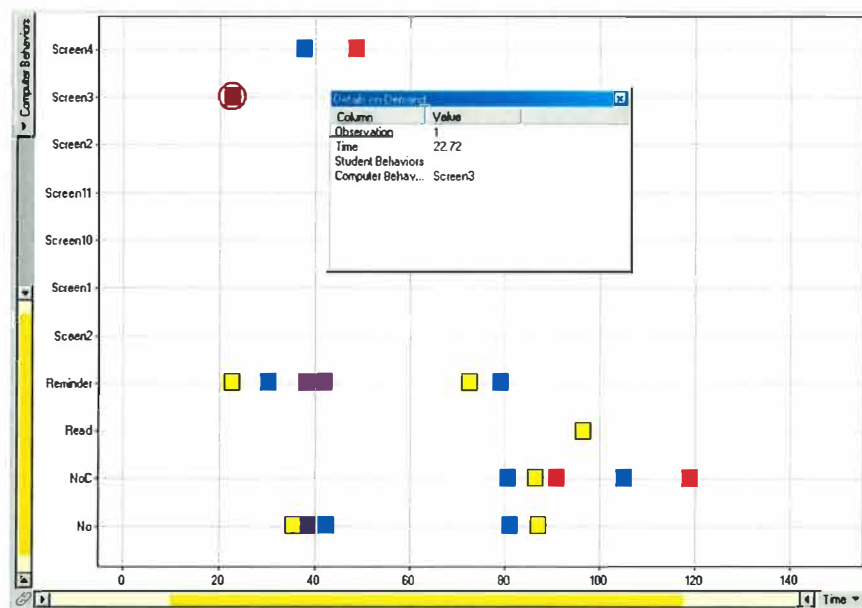


Figure 30: Spotfire "Details-on-Demand" internal frame

Results of the Spotfire analysis

As we already know that our system will be using multiple views, we can take advantage of this analysis to take into account the most useful features of Spotfire.

First of all, the analysis of the general principles ruling Spotfire has given some information about the presentation of the different views. We have seen that displaying views side-by-side facilitates the investigation by illustrating simultaneously different aspects of the same data. On the contrary, a multitude of views displayed at the same time adds complexity, reduces the visibility of each view and makes it difficult to mentally switch between views and to take the best of each of them. A first recommendation concerning the design of the interface will thus be to limit the number of views that are simultaneously displayed. However, limiting the number of views imply the presence of a very simple and efficient mechanism to remove and replace views by others.

A second interesting point developed by Spotfire is the brushing interaction. This means of interaction provides a direct overview of all items of interest on all displayed views and facilitates therefore the context switching from one view to another. However, the real interest and efficiency of such an interaction tool definitely depends on what is displayed on the view. Furthermore, highlighting items on a table, a 2D or a 3D view requires different mechanisms that must be consistent and that cannot decrease the usability of the view. That is why we refer to section 5.4 for a further discussion on whether or not to include this feature or another means of interaction in our application.

The illustration of our functional requirements with Spotfire has also led to a set of useful considerations for the design of multiple view systems in general.

First, we have seen the importance of an access to a wide panel of data sources. But, as it is stated in section 4.3.4, we will not deal with this problem in this thesis.

Then, we have seen the benefits of a dynamic update of the views displayed when selecting and filtering information. In the field of 3D virtual universes, El ansari and Vanbrabant outlined in [El ansari and Vanbrabant, 2001] the importance for future works on the subject to give applications a fully interactivity in the direct manipulation of the graphs that are displayed. This interactivity does not only involve a navigation tool but must also provide the user with the ability to modify the virtual world. With this interactivity, the user can get an immediate response in the system without having to start a new Select-Filter-View process. From there, we can derive a second recommendation for the design of our system: It should be possible to get an immediate update of the views displayed when performing the points 3 and 4 of the generalised task model in section 4.2.5, i.e., the selection of the observations and the filtering of the data.

The visualisation techniques illustrated above (Pie Chart, Histogram, Scatterplot, Scatterplot 3D and Table) are, on the contrary, not relevant for us. These views are not at all designed for the visualisation of usability data and we will therefore only consider them a source of inspiration for the design of the visualisation metaphors in section 5.4.

The last recommendation that emerges from this analysis is related to the "view details on demand" task. Displaying on each view complete information on each item displayed is in most cases impossible. It could be therefore very useful to link each item to its complete description. Like for the brushing interaction, we will decide further what is the best way to get complete information on items displayed.

The other tasks described (The "zoom", "undo/redo", "save view" and "print view" tasks) do not provide any original characteristics for our point of view or are simply not available.

5.3.4 The interface

Introduction

In this section, we design each component of our interface with regard to the requirements and recommendations above and using some other general guidelines for the design of application, most of them coming from [Fowler, 1995], [Microsoft, 1992] and [Galitz and Wilbert, 1989]. For each of the components described, we illustrate our reflection by some screenshots of our program.

Space allocation in the main window

The main window traditionally contains the following elements:

- The title bar
- The menu bar
- The control bar
- The window body
- The status bar

The title bar

The title bar contains the control menu at the left, minimise and maximise/restore buttons at the right, and the title of application name in the middle. If the application has files, then the name of the opened file should be appended to the application name. Otherwise, the title of the window is the application name [Fowler, 1995].

It is thus time for us to choose a name and a logo for our application. The name chosen is "AlgoaGraphics" with reference to the former name (Algoa Bay) of Port Elizabeth (South Africa) where we have spent about five month working on this project. The corresponding logo is shown on Figure 31.

As it is required in [Fowler, 1995], when a project is open, the title will also contain the complete path of the opened project (i.e. configuration file).

The menu bar

The menu bar is situated at the top of the window, just below the title bar. The content of this bar will be defined further according to the different functions the application will support.

The control bar

Galitz and Wilbert [Galitz and Wilbert, 1989] define the control bar as an array of options designed to be accessed quickly and repeatedly. It is generally used to hold toolbars, palettes, rulers and ribbons. A control bar can be fixed to the edges of the main window or float above it in a dialog box. A well-behaved application should let the user:

- Hide or close all components of the control bar, so that users have more working place when they need it.
- Access all controls from menus or the keyboard as well as the mouse

This application will use a toolbar containing the most frequently used options. This toolbar will be fixed to the top edge of the window, below the menu bar. The appropriate content of this bar will be discussed further, but we can already say that:

- All options in this bar will have a corresponding option in the menus of the menu bar
- An option of the menu bar will allow the closure of the toolbar

The window body

The previous sections led us to the adoption of multiple views combined to the dynamic update of the views displayed when performing the selection and filtering tasks. We have also seen that these last two tasks were fundamental for the effective visualisation of usability data. They provide a way to quickly focus on the items of interest (observations, subjects, behaviors, time period) and to quickly switch from one investigation point to another. That is why, gathering these two functions in one simple, removable internal frames (cf. Spotfire "Query Devices" internal frame) do not seem to be appropriate for the manipulation of usability data. When using Spotfire, we besides noticed that the "Query Devices" internal frame stayed maximised during the whole testing session. In order to provide constant access to these functions, we will thus use a fixed panel located at the bottom of the window body. This panel will contain all the necessary information to see complete information on observations, select observations and filter data (tasks 2, 3 and 4 of the generalised task model), while the remaining space (the middle of the window body) will be dedicated to the visualisation of the views selected.

The status bar

The status bar contains, at minimum, window identification. It can also contain status messages, running commentary on the buttons and menu options that the user highlights.

Since our application is composed of a unique window, the status bar would be useless and above all space consuming. It is therefore not available.

Figure 31 shows the space allocation for the main window.

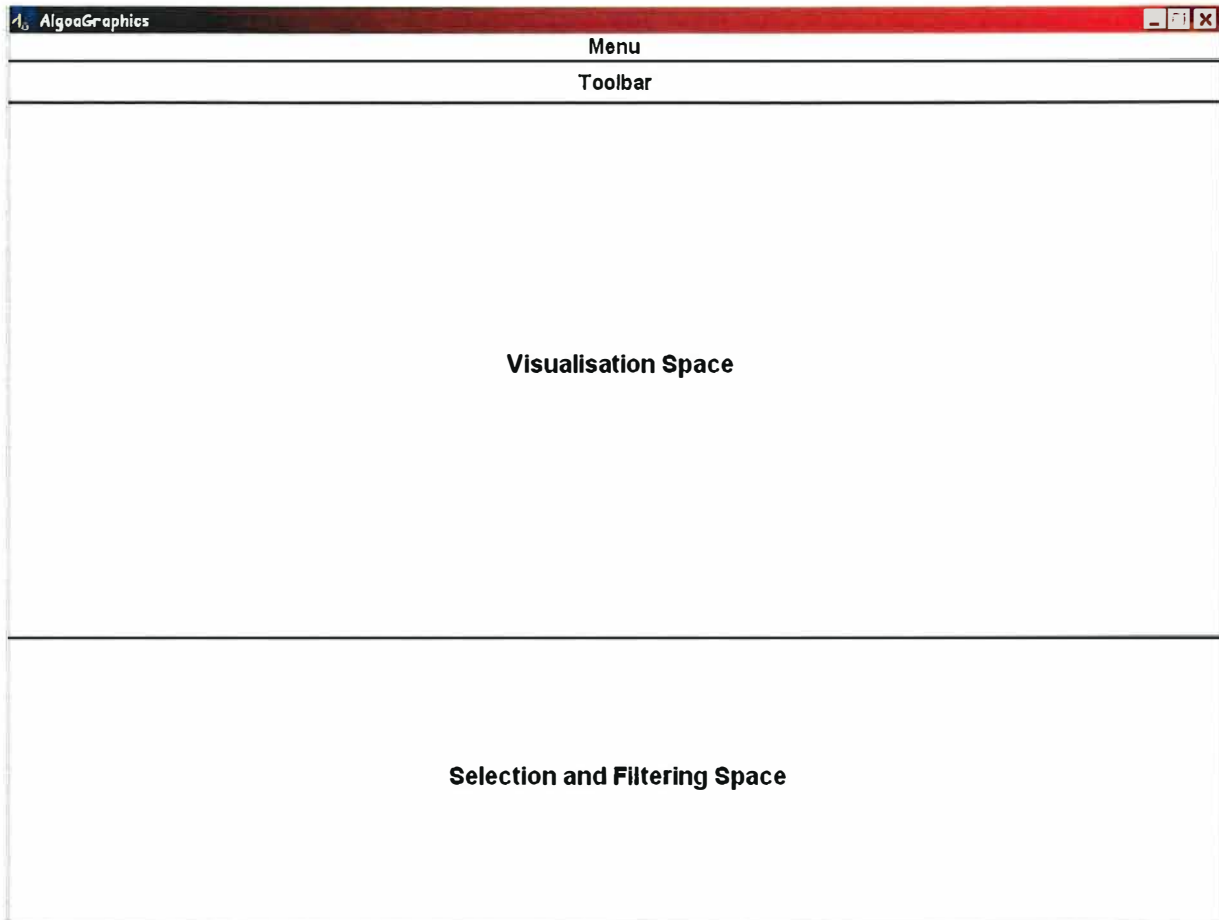


Figure 31: Space allocation for the main window

Selection and filtering panel

Content of the panel

This must allow the following tasks to be performed:

- The consultation of the information related to each observation, including information on:
 - The observation itself
 - The user
 - The device
- The selection of the observations, directly or on the basis of independent variables related to the users involved in the observations, i.e. the gender, the age, the language and the expertise

- The filtering of the data including:
 - The selection of the subjects (user and computer)
 - The selection of the behaviors performed by these subjects
 - The selection of the time period

Though the overall density of a window should be between 25 and 30 percent, i.e., 70 to 75 percent of the window should be empty space [Galitz, 1989], one should keep in mind that expert users prefer denser displays because more information per screen means fewer computer-related operations. We will thus attach very little importance to these percentages and try to organise this space-limited panel in the best way to allow the simultaneous access to the three tasks mentioned above.

The selection of the observations

The selection of the observations should ideally allow one to:

- Select all observations (default selection)
- Select observations on the basis of the independent variables
- Select directly some of the observations

As these three subtasks are mutually exclusive, we have chosen to use three radio buttons, each of them displaying the appropriate panel to perform the corresponding subtask.

The selection of all observations is shown on Figure 32. It simply displays the list of all observations related to the opened project in the panel below.

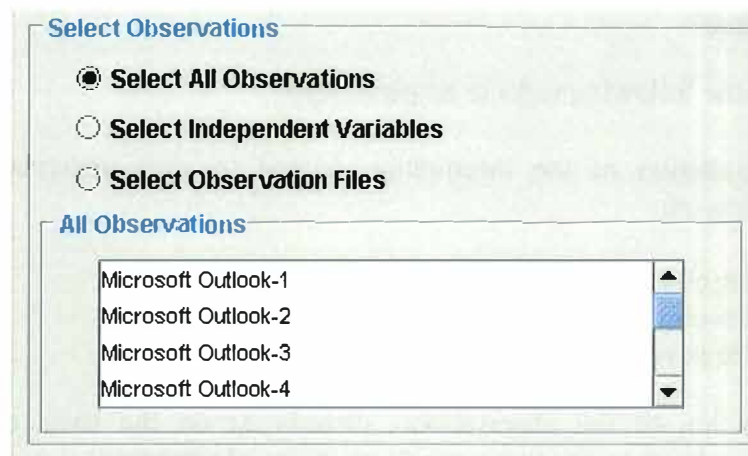


Figure 32: Selection of all observations

The selection of observations on the basis of independent variables is shown on Figures 33 to 38. The corresponding panel below is divided into five tabs. The first tab gives an overview of the observations selected. This selection corresponds to the current value of the independent variables (cf. Figure 33).

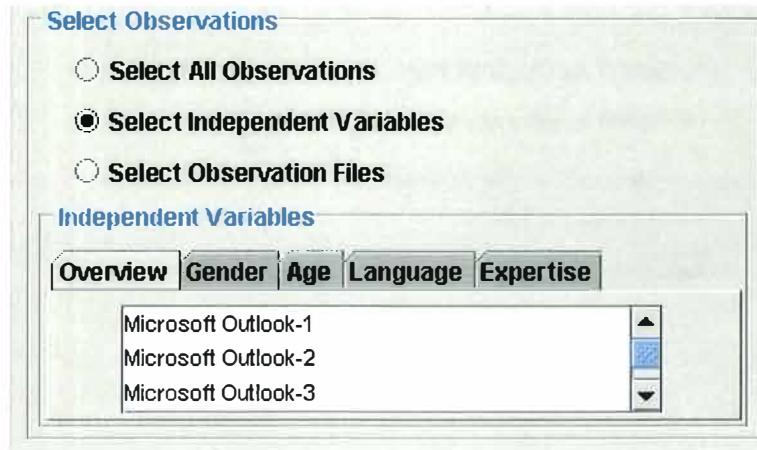


Figure 33: The "Overview" tab

The last four tabs allow the handling of the four user related required independent variables:

- The first one provides a way to select the desired gender of the users involved in the observations through two checkboxes (male and female). These checkboxes are initially both checked. (cf. Figure 34)

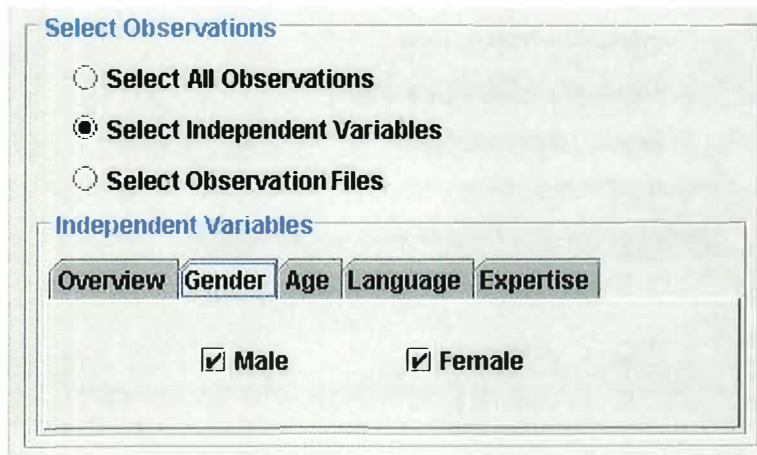


Figure 34: The "Gender" tab

- The second one is dedicated to the selection of an age period. It provides two radio buttons to decide whether or not to include the age in the selection process and two spinboxes to specify the age limits. We can further notice that the default selected age period is between 0 and 99 (cf. Figure 35)

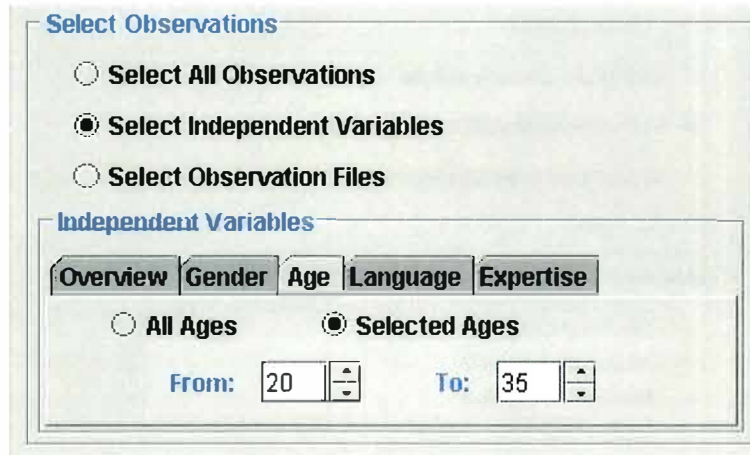


Figure 35: The "Age" tab

- The third one allows the selection of the desired languages. It also provides two radio buttons in order to include or not the language in the selection process. The choice of the languages can be made through a list of checkable items. Because of the limited available space, this list only displays one row at a time. All languages are initially selected (cf. Figure 36)

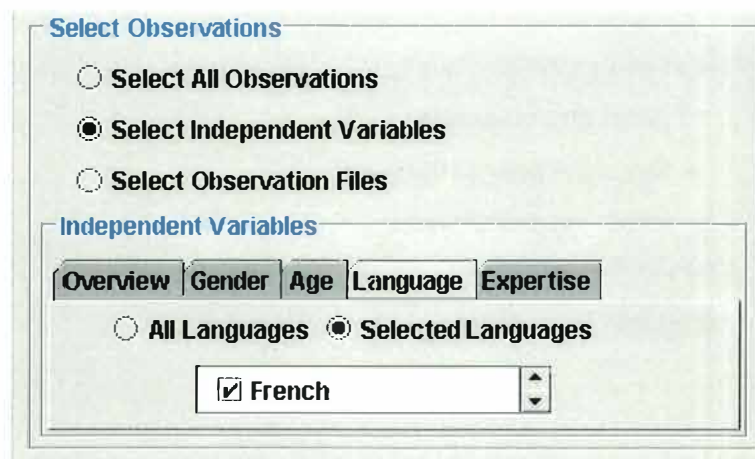


Figure 36: The "Language" tab

- The last tab provides a way to select the desired experience of the users involved in the observations. The experience can be either "novice", "medium" or "expert". The choice is possible through three checkboxes initially checked (cf. Figure 37)

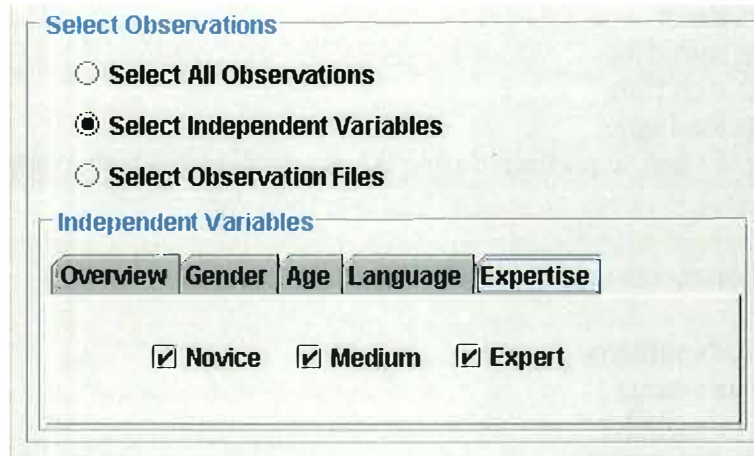


Figure 37: The "Expertise" tab

Finally, the direct selection of some of the observation files is shown on Figure 38. This can be done by using a system of two lists, one containing the non-selected observations, the other containing the selected observations. The transfer of an item from one list to another is possible using two one-way transfer buttons. We can further notice that the lists are automatically sorted on the observation names and that none of the observations are initially selected.

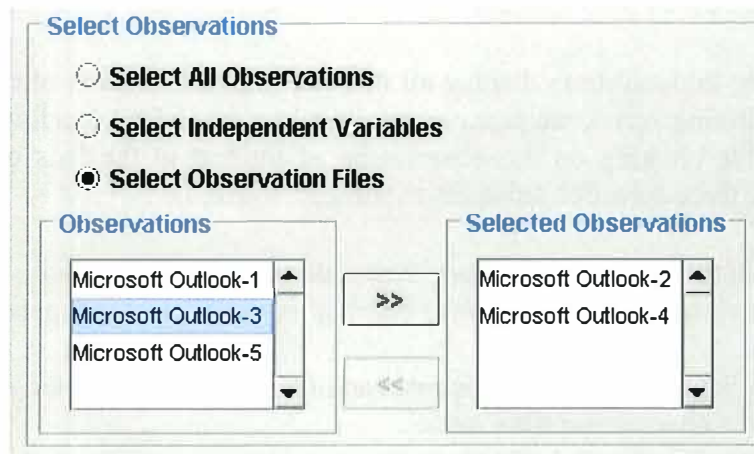


Figure 38: Selection of some of the observations

The consultation of information about observations

A complete description of an observation includes:

- Information on the observation itself, i.e.:
 - Its name
 - Its start date
 - Its start time
 - Its total time
 - If the task to perform during the observation has been completed or not

- Information on the user involved in the observation, i.e.:
 - One's surname
 - One's name
 - One's gender
 - One's language
 - One's expertise

- Information on the device involved in the observation, i.e.:
 - Its type (computer, cell phone, PDA, ...)
 - The media available (text, image, audio, video)
 - The screen size
 - The number of colours
 - The interaction tools available (keyboard, mouse, ...)
 - Other features peculiar to the device

As it is basically impossible to display all this information for each observation in the selection and filtering panel, we propose to put it in a secondary window that could be opened by double clicking on the observation of interest in the lists of observations available for the three selection subtasks mentioned above, i.e.:

- The list of all observations in the "Select all observations" panel
- The overview of the observations selected in the "Select independent variables" panel
- Both the list of selected observations and the list of non-selected observations in the "Select observation files" panel

Figure 39 shows an example of the information displayed when double clicking on an observation. When the content of a field has no length constraint and is longer than the window width, a "More..." button appears in the left edge of the window to get the complete content displayed.

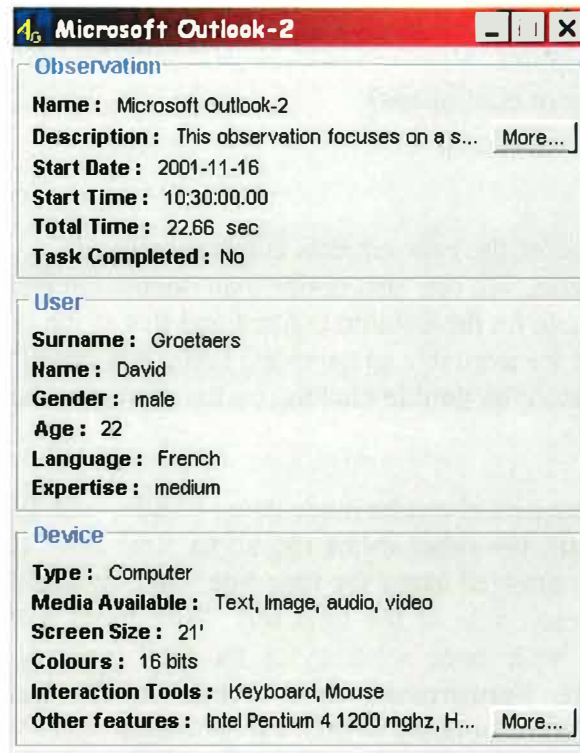


Figure 39: The complete description of an observation

The filtering of the data

The "filter data" task includes:

- The selection of the subjects
- The selection of the behaviors
- The selection of the time period

The selection of the subjects is made possible using two checkboxes, one for each subject (cf. Figure 41). These ones are obviously initially selected.

Behaviors can be selected through two radio buttons and a table (cf. Figure 41). The two radio buttons allow one to choose between including all behaviors or only the selected ones. The table contains checkable items including all the necessary information to decide whether or not to select the behavior, i.e., for each behavior:

- Its name
- Its textual description
- Its type (discrete or continuous)
- The corresponding subject

Of course, checking one of the two subjects automatically add all the related behaviors to the table and vice versa. We can also notice than double clicking on a column header of the table sorts the table on the column content and that if the textual description of a behavior is longer than the available space in the table, it is possible to get the complete description of this behavior by double clicking on the corresponding description cell.

The selection of the time period can be made using a slider with two knobs, the first one for the lower time limit, the other is for the upper time limit (cf. Figure 40). These knobs can of course be dragged along the time line, but they can also be moved using a couple of buttons on each side of the time line. With these buttons, it is possible to select the time period with more accuracy as the total times of the observations are theoretically unlimited. Furthermore, any change in the position of the knobs dynamically updates a corresponding textual value on each side of the slider.

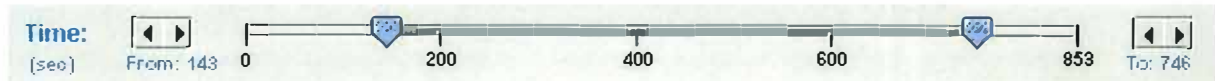


Figure 40: The selection of the time period

The selection and filtering panel: overview

All the components described above are illustrated on Figure 41. The time line is situated at the top of the panel to benefit from the total width of the window. The selection panel is situated on the left side and the filtering panel on the right side.

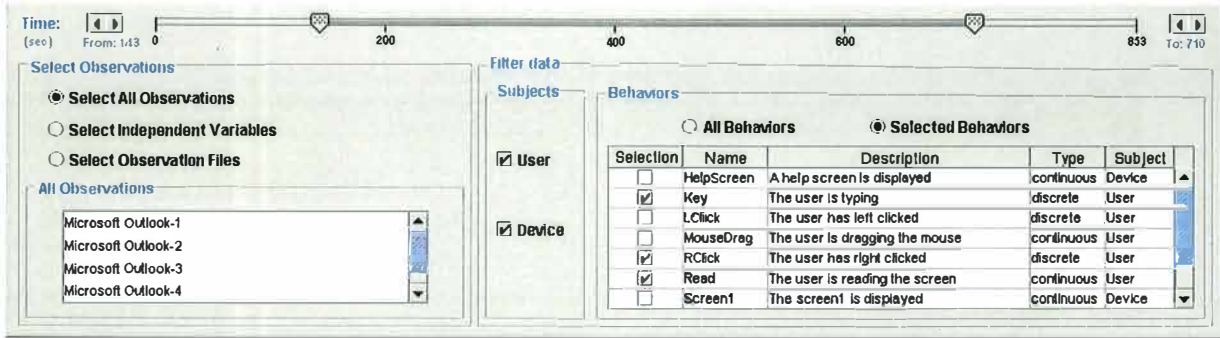


Figure 41: The selection and filtering panel: overview

Visualisation panel

We have already opted in section 5.3.2 for a side-by-side presentation of the views. Given the available space remaining for the display of the views and as the Spotfire analysis has shown that their number should be reduced to permit the effective use of each of them, we will limit the number of simultaneously displayed views to a maximum of three.

Furthermore, using internal frames (cf. Spotfire) does not seem to be appropriate for our needs. With a maximum of three views, we think that providing a fixed size for the display of one, two or three views is better than a direct, time consuming, manipulation (open, close, resize, minimise, maximise) of three distinct frames. In that case, the system will have to provide a way to quickly and easily increment and decrement the number of displayed views. We refer to the menu section for further discussion on the manipulation of the views.

Figures 42 to 45 show the visualisation panel with respectively one, two and three simultaneously displayed views. The three-view panel is initially displayed.

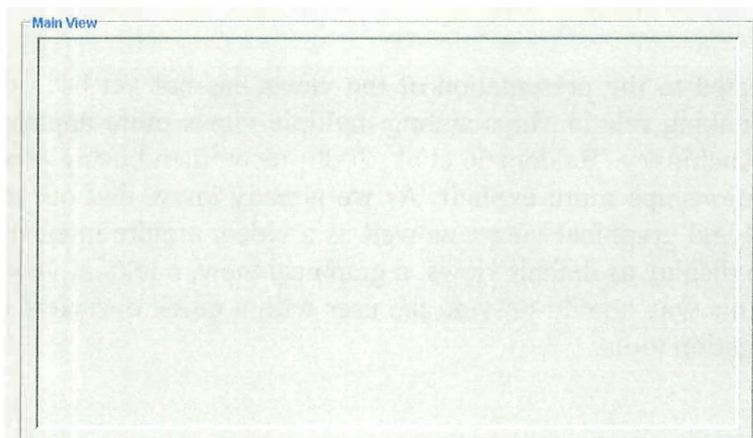


Figure 42: Visualisation panel with a unique view

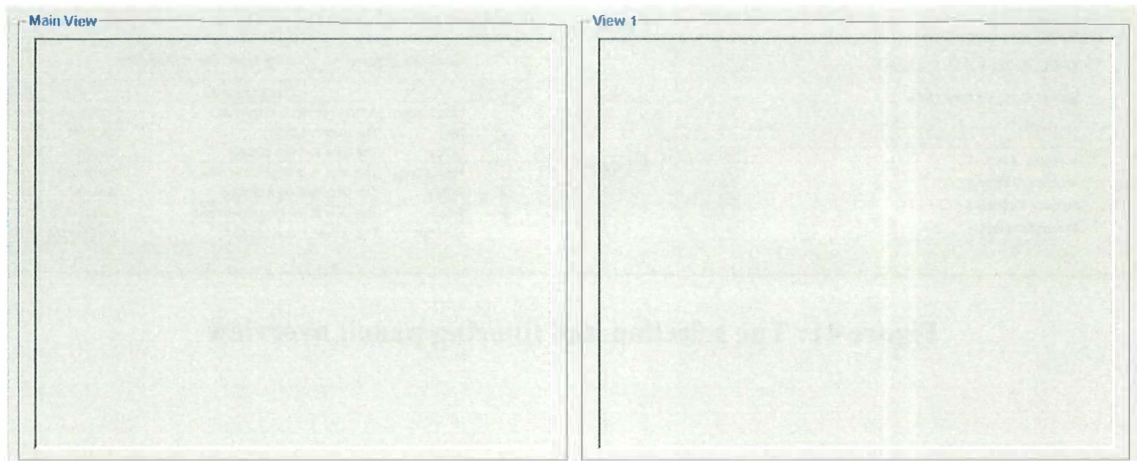


Figure 43: Visualisation panel with two views

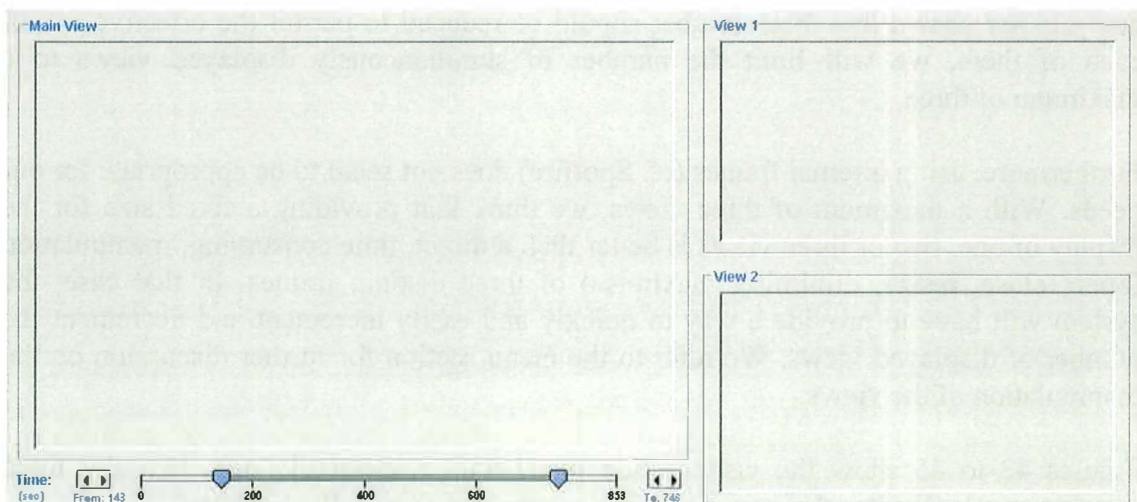


Figure 44: Visualisation panel with three views

A last point related to the presentation of the views has not yet been discussed. This point is about making relationships among multiple views more apparent. Baldonado, Woodruff and Kuchinsky [Baldonado et al., 2000] recommend using perceptual cues to make these relationships more explicit. As we already know that our application will use both textual and graphical views, as well as a video, a quite interesting and simple cue would be to display as default views, a graphical view, a textual view and the video viewer. Doing this way should provide the user with a quick overview of all available kinds of investigation tools.

Menus

The menu bar

From the generalised task model in Chapter 4, and the recommendations made in this chapter, we can derive the main actions that the menu bar should allow. These are listed below:

- Open a project (i.e. a configuration file)
- Close the current project
- Undo the last action performed
- Redo the last action cancelled
- Save the main view in a common image file format (bitmap)
- Print the main view
- Hide/Display the toolbar
- Switch between the one, two and three view displayed panels described in the "Visualisation panel" section
- Change the graph contained in one of the displayed views
- Get one of the views in a full screen window
- Get a help screen

These options are organised as follows. For each of the four standard menus below, we give a list of the contained options and submenus.

- The File menu
 - Open project
 - Close project
 - Save view...
 - Print View...
 - Exit
- The Edit menu
 - Undo
 - Redo
- The View menu
 - Toolbar (checkbox)
 - Views (Radio buttons to chose the number of displayed views)
 - 1 View
 - 2 Views
 - 3 Views
 - Graphs
 - Main View: list of all possible graphs (to be defined)
 - View 1: list of all possible graphs (to be defined)
 - View 2: list of all possible graphs (to be defined)

- Full Screen
 - Main View
 - View 1
 - View 2

- The Help menu

The help function is not available. We refer to the content of this chapter for a description of the application

In order to avoid including a multitude of screenshots in this section, we refer to the use of the application to get an illustration of the menus and submenus described above. The appropriate icons, mnemonics and accelerators are chosen with reference to Microsoft Windows products and follow some general guidelines coming from [Fowler, 1995].

The pop-up menu

A pop-up menu is defined as a secret, expert menu that appears when the designated mouse button is clicked (usually the right one) on an active area in an application window [Fowler, 1995]. We mainly use the pop-up menu illustrated on Figure 45 to facilitate the handling of the views, as it is required in the "Visualisation panel" section. This menu will appear when right clicking on a view and will contain the following options:

- Save
- Print
- Graphs (list of all possible graphs allowing to quickly switch from one investigation tool to another)
- Full Screen

Figure 45 shows the pop-up menu being displayed.

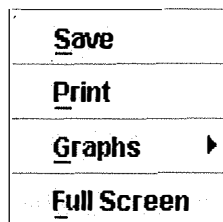


Figure 45: The pop-up menu

Toolbar

The toolbar shown on Figure 46 contains icons for a quick access to the most frequently used options of the menu bar, i.e. in order of appearance:

- Open project
- Save View...
- Print View...
- Undo
- Redo
- 1 View
- 2 Views
- 3 Views

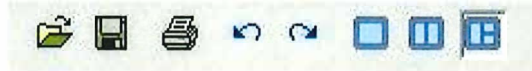


Figure 46: The toolbar

Interface overview

Figure 47 shows an overview of the whole interface.

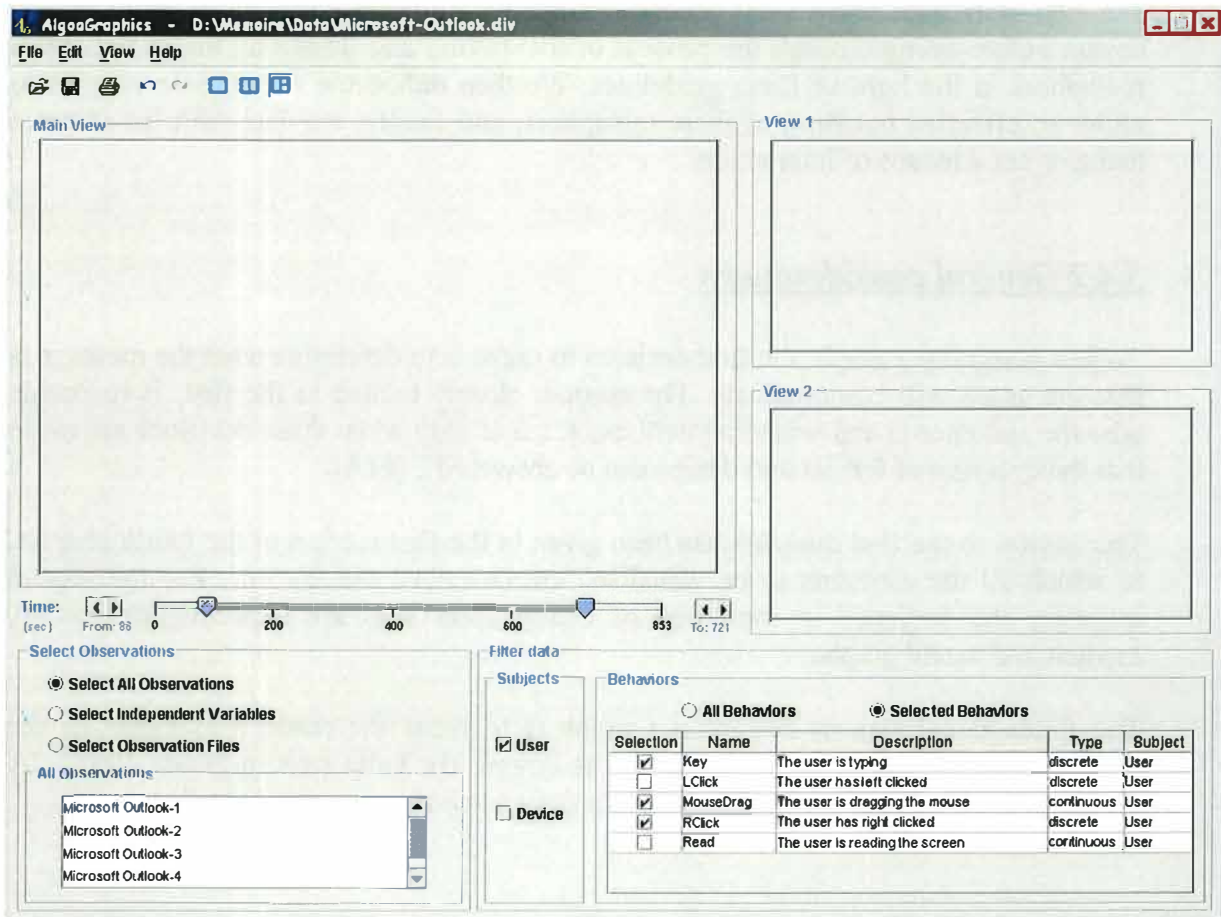


Figure 47: The interface: overview

5.4 Design of the visualisation metaphors

5.4.1 Introduction

The main goal of this section is to tackle the last issues that have not yet been discussed. These are listed below:

- Identify the appropriate number of visualisation metaphors to use with reference to the fourth rule for the use of multiple view systems: the rule of parsimony.
- Identify a set of visualisation metaphors and discuss their design
- Decide whether or not to use a brushing interaction or another means of interaction
- Satisfy the "View details on demand" task by displaying complete information on each item displayed on the views

We first start with some very general guidelines outlining concepts of good graphs design before going through the process of identifying and designing the visualisation metaphors in the light of these guidelines. We then define the appropriate viewers to allow an effective handling of these metaphors, and finally, we deal with the issue of using or not a means of interaction.

5.4.2 General considerations

“When designing a graph, the first decision to make is to determine what the message is that the graph will communicate. The second, closely related to the first, is to decide who the audience is and what they will expect. It is only when these decisions are made that the question of format and design can be answered”, [EIA].

The answer to the first question has been given in the first section of the fourth chapter, in which all the elements to be visualised are described one by one. For the second question, the audience is composed of expert users who are expecting interactive, explicit and useful graphs.

The fundamental task in designing a graph is to focus the reader’s attention on the graph’s data and on its message, not on its design. As Tufte says in [Tufte 1983], “A well-designed graph displays the minimum design and the maximum data”.

The following are general principles for a unique graph given in [Tufté 1983]:

- Above all else show the data
- Avoid distorting what the data have to say
- Encourage the eye to compare different pieces of data
- Minimise the number of mental operations: fewer operations take less time and produce fewer errors
- Keep the data-ink ratio (amount of ink to graph the data divided by the total amount of ink in the graph) as high as possible. The less information shown in the display, the worse it is. So it is important to minimise amount of ink that doesn't depict actual data points or the relation between them.

If multiple graphs are displayed in the same time, which will be our case, then, according to Tufté, there are four specific concerns that should be kept in mind:

- Coding variables
Build graphs so that quantitative variables are along X-axis
- Consistency
Be consistent between graphs in terms of coding data
- Highlighting differences
For related material in different graphs, highlight differences from graph to graph, either prominently in legend or in the symbols
- Distinctive legends
Legends of similar graph must highlight distinct features

5.4.3 Design Criteria

This section discusses basic design features such as titles, frames dimensions, X- and Y-axes scales, information messages, and colour.

Title

Every graph needs a clearly worded and concise title that answers the question of the data illustrated (“what”).

In graphs, unlike as in data tables, the unit of measurement is not in the title because it is displayed on the axis.

The title of each graph will be displayed at the top of the view, next to the view name (i.e. Main View, View 1, View 2).

Frame Dimensions

Graphics tend towards the horizontal rather than the vertical. The primary reason is that “horizontally stretched time series are more accessible to the eye” [Tufte 1983]. A pleasing shape is called the “golden rectangle” in which the ratio of the longer axis to the shorter axis is 1.6 to 1. The graphical perception research has found “a mild preference for proportions near the golden rectangle”. Given that the golden rectangle is a tendency and a preference, Tufte concludes:

- “If the nature of the data suggests the shape of the graphic, then follow that suggestion.”
- “Otherwise, move towards horizontal graphics about fifty percent wider than tall.”

Scale

In a time graph, the horizontal scale (X-axis) always represents time. Demarcations of time intervals must be proportional to the length of the time represented. The vertical scale (Y-axis) displays the units of measurement, an average or a quantity, of the data.

Logically, there is only one vertical scale, because the purpose of the scale is to provide the background against which the data shown in the graph are judged. Multiple vertical scales may be used only if the scales are linearly related to one another.

Scale Intervals and Tick Marks

In designating scale values, 5 or multiples of 5 are frequently used in preference to multiples such as 3, 6, 7, 9, 11, and so on. Scale divisions for both axes are indicated by major tick marks. Major tick marks usually are outside the axes and are labelled with the scale values. In certain scales, the use of minor tick marks is a good idea, particularly on a Y-axis with a wide scale range (i.e., 0 to 5,000 with intervals of 1,000). Minor tick marks, though not labelled, assist the user to read the data more quickly and accurately. The minor tick mark divisor should be a number that the reader can instantaneously divide into the major tick mark number. If the reader has to think about the division (i.e., 6 into 1,000), then the minor tick marks are counterproductive.

Scale Labelling

For clarity, in addition to the units designated on the scale, both the horizontal and vertical axes need explanatory labels for the variables represented. The unit of measure is customarily also specified in the explanatory label.

The Y-axis label normally is placed parallel to and reading inward to the Y-axis because the Y-axis label refers to the Y-axis scale, not the top frame line. There are different conventions for placing this label to avoid having to read it sideways: it is possible to put the unit in the graph title, or put it on the top frame line. The generally accepted, and most widely used, practice is to place it parallel to the scale to which it refers. In situations where there are multiple graphs on a page and all the graphs have the same scale, the Y-axis label can be placed in the title subheading.

It is a good idea to avoid abbreviations and acronyms, and if they must be used because of space limitations, then they need to conform to the standards.

We can notice here that it is often a good idea to make marks or labels as small as possible, but while being still clear.

The Zero-Base

As a rule, the vertical scale starts at zero. Otherwise, the relative importance of changes in levels is hard to assess and comparison is difficult, or an insignificant change can be made to look like a major change.

Grid lines

Horizontal grid lines are often necessary. When needed to guide the eye, they should be kept to a minimum, evenly spaced, and comparatively light in contrast to the data lines. If grid lines are not light, they may overpower or hide the data lines, and thus obscure or distort the presentation. So it is better to avoid busy grids, and, more generally, to concentrate on the data rather than on the data containers. Vertical gridlines, however, are often helpful. It is recommended that they be used, for example, to separate a change in the data series (i.e., the method of sampling was changed).

Messages

If they do not clutter the graph, short informational messages are very useful and can be added to the graph. Messages may provide additional context to the data, emphasise a particular point, and/or explain ambiguous or possibly hidden features in the graph. Messages are particularly useful in explaining the data (and even the graph format) in graphs that are not easily comprehensible on account of the subject matter complexity.

Colours

The appropriate use of colours is very important for effective visualisation, and there are some general principles of effective colour use and interaction. The purposes of colours are to differentiate and to encourage comparison [EIA]. First, as described in [Brown, 1995], it is important to know that the human eye is more sensitive to certain colours than to others. For example, the eye is least sensitive to blue, and therefore it is very difficult to distinguish between two objects that differ only by the amount of blue. Blue is actually a good colour for background or large areas. Similarly, red and green are both more effectively used in the centre of the display than in the periphery.

In "Envisioning Information" [Tuft, 1990], Tuft suggests that authors should not use pure, bright or very strong colours "unrelieved over large areas adjacent to each other". This creates "noise" that distorts, blurs, or covers the visual impact of lines, bars, strata, map contours, etc., which represent the data. For example, "if you put bright blue next to either bright green or bright yellow, the edges will be blurry. This problem can be avoided by increasing the difference in brightness between the two colours in such a colour pair." Similarly, "the use of bright red in combination with either bright green or bright blue causes an appearance of vibration and possible afterimages at the edges where the colours meet." Furthermore, these vivid combinations are pretty tiring for the eyes, because they must continually refocus to see each colour.

In addition to these obvious colour combination clashes, one has to remember that the same colour can look very different when it is used against different backgrounds, and if the colour areas are different sizes. Pure, bright and very strong colours should be used sparingly in graphs, and further to this, the use of such colours against a black background produces a "1+1=3" effect, a form of noise where the colours are at "visual war" with the information in the data display [Tuft 1990].

For the choice of the best colour combinations, we will refer to the ones exposed in [Brown, 1995] p132, that describes the thin or thick lines, panels or text colours which are most appropriate for some background colours.

The following are additional guidelines on the use of colours advocated by Tufte [Tufte, 1990]:

- Use only colours that enhance the data presentation. Do not use colour for graph frames, large unnecessary symbols, or titles, for example
- Layering with colours is often effective
- The colour scale should be proportional to the data scale. For example, big differences in colour should not be used to represent small quantitative changes
- Colour grids are a form of layer which provides context but which should be unobtrusive and muted
- Pure bright colours should be reserved for small highlight areas and almost never used as backgrounds.
- Use colour as the main identifier on computer screens as different objects are often considered the same if they have the same colour regardless of their shape, size, or purpose
- Note that surrounding colours can make two different colours look alike, and two similar colours look very different
- Subtle shades of colour or grey scale are best if they are delimited with fine contour lines
- Be aware that 5-10% of people are colour blind to some degree (red-green is the most common type followed by blue-yellow, which usually includes blue-green)
- Use colours or hues that are distinguishable without one colour overpowering the other(s). Tufte recommends using the "smallest effective difference." The "visual move" from one hue (or colour) to the next is as small as possible but is still distinctive and clear. This allows for more differences and, hence, data to be displayed.

Three-dimensional features

We have to wonder for each graph whether to introduce 3-D or not. Though there may be good reasons for using 3-D displays, as said in [EIA], “3-D graphs can be useful for displaying three related, continuous variables, but to add a non-data spatial dimension often introduces distortion and ambiguity that distract from the clarity of the presentation“. According to [Beaton], it is necessary to be very cautious when introducing 3D. 3D representations in 2D display plan are inherently ambiguous with regards to depth and distance, and not useful for precise value readings.

[Brown, 1995] affirms that “It is clear that every user with an application involving multidimensional data can benefit from interactive 3D graphics [...]. However, despite the benefits of incorporating 3D graphics into applications, users and developers may still be put off by the challenge of working in 3D space, as well as the development expense.”

For all these reasons, we will have to be very careful in deciding to introduce a third dimension when designing the metaphors in the following section.

5.4.4 Identification and design of the visualisation metaphors

Next step consists in wondering what data will be simultaneously present on a particular chart or table and what format is best adapted for the visualisation of this data.

As the rule of parsimony recommend to use multiple metaphors minimally, we will try in the following to reduce at the most their number by gathering similar information in a single structure. We will also keep in mind the "View details on demand" requirement and we will explain for each metaphor how it provides the user with the ability to consult complete information on items displayed.

As a starting point for this discussion, we go back to Chapter 4 and the list of elements we want to visualise. Here is what we want to visualise:

For each observation separately or in comparison with others:

- If the participant has completed the task successfully or not
- The total time taken to complete the task
- The sequence of actions and the time spent for each action in the completion of the task
- The comparison between the sequence of actions followed by the user and an optimal sequence of actions
- The number of times each behavior has been observed in the completion of the task
- The total time spent per behavior in the completion of the task
- The number of times each behavior has been followed or preceded by another in the completion of the task
- The video of the observation if it is available

For all observations taken together:

- The completion rate or percentage of participants who completed the task correctly
- The mean task time or mean time taken to complete the task
- The completion rate efficiency or completion rate/mean task time
- The number of times each behavior has been observed across observations
- The total time spent per behavior across observations
- The number of times each behavior has been followed or preceded by another across observations

The following sections illustrate each of the seven metaphors chosen to visualise these elements.

Temporal sequence of behaviors

First graph allows one to visualise:

- The sequence of behaviors and the time spent for each behavior in the completion of the task

This graph makes time intervene, and needs an axis representing time. As explained in section 5.4.3, the time should always be represented by the horizontal scale. This scale's label logically is time, and displays the time in seconds. As recommended in section 5.4.3, we have decided to only use multiples of five for the tick marks values. A maximum of five tick marks are displayed at the same time, not to overload the chart, and to keep their values readable. Actually, only certain values may be displayed. These are either "5, 10, 15, 20, 25", "10, 20, 30, 40, 50", or "20, 40, 60, 80, 100", or, any of these values multiplied by 10.

We want to show the sequence of behaviors performed by the subjects during one or more observations. To do that, we represent each sequence of actions as a bar, which is plotted horizontally against time. This means that the different observations are represented on the vertical axis.

It then becomes obvious that this chart will take the shape of a bar chart, in which bars are plotted horizontally against time, and in which they represent a sequence of behaviors performed during the task. The length of a bar must be proportional to the duration of the sequence of behaviors. The bar is divided in several rectangles of different width, each of them representing a behavior. Each rectangle composing this sequence is to be identified by a colour, and separated from the preceding one and the next one by a vertical line.

The choice of colours for distinguishing the behaviors has been preferred to other solutions such as noting the behavior names inside the rectangles. This one had different disadvantages among which a slowing effect on the program, making the zoom and translation much less fluid. Another problem was the fact that sometimes, in the case of short rectangles, the available space for writing their name was really small and made their reading impossible. It was then necessary to zoom, though colours stay visible even in very small areas and thus make the behavior's identification nearly immediate. Another reason for choosing colours is the general overview that they quickly allow when looking at the graph. Indeed, one can immediately see which are the dominant colours and the less visible ones, and therefore get a quick opinion about an observation. For example, if a help screen is very often displayed, and for a pretty long time, this might be a sign that investigation is required.

We can still notice about the colours that the technique we have developed in order to choose them is designed in a way which allows one to select the most different ones, in function of their number. Furthermore, we have tried to make the three basic colours (red, green, blue) intervene in the same proportion in the creation of the different colours. This means that the chart should make appear very different colours, and not only hot or warm colours.

A few other comments still have to be made about the way we have chosen to display the data. Indeed, since this graph is intended to display data related to the behaviors and observations times, we have been confronted to the question of this data's semantics. Actually, as explained in Chapter 4, two subjects are involved in an observation: a user and a device. Each of them performs behaviors, that can be either continuous or discrete. Thus, just displaying the sequence of all behaviors, irrespectively of the subject by which they were performed, would be insufficient since the real duration of a continuous device behavior, for instance, is not the duration extending from its start time until the start time of the next behavior, whatever its kind. We consider it is the duration between its start time, and the start time of the next continuous device event, since a user event does not necessarily change the device current state. The best example to justify this is the commonly used read behavior: if a user reads, it does not change, for instance, the screen displayed. Therefore, the duration displayed is calculated in a different way in function of its type (continuous or discrete), and of the subject by which it was performed.

- Any user behavior's duration is calculated from its start time up till the next performed action begins, whatever its type and the subject that performed it. Indeed one can reasonably suppose that the occurrence of a device behavior normally interrupts the current user behavior. Of course there might be exceptions, but we had to make choices in this matter.
- However, the contrary is not true, and the calculation of a device behavior's duration depends on its type:
 - A continuous device behavior's duration is calculated from its start time up till the start time of the next continuous device behavior
 - A discrete device behavior's duration is obtained from its start time up till the start of the next behavior, whatever its type and the subject that performed it. This seems appropriate since the behavior is discrete, and normally has no real duration.

The exception concerns the only continuous device behaviors. This means that when selecting a behavior, the displayed duration might not correspond to the behavior's size in regard of the time axis. This is the case for the continuous device behaviors. It thus appears necessary to display a second bar, dedicated only to those ones. This bar will be situated just under the "normal" one, and will have the same size. This means that the normal sequence of behaviors can be directly compared with the sequence of continuous device behaviors. The display of this new bar has the advantage of allowing one to have a quick and easy overview of, for instance, which screen is displayed while the other behaviors are performed. This fact can obviously help identifying and understanding the potentially existing usability problems more easily. Of course this data might be available through the selection of the desired behaviors in the selection panel described above. But the interest is to have the possibility to compare both sequences, and to visualise them both in the same time.

We can add that the fact of calculating a discrete behavior's duration may seem strange. But this chart is aimed at displaying data about time, and since we have to represent those behaviors in the graph, it is the best way to achieve this.

Figure 48 shows an example of the temporal sequence of behaviors:

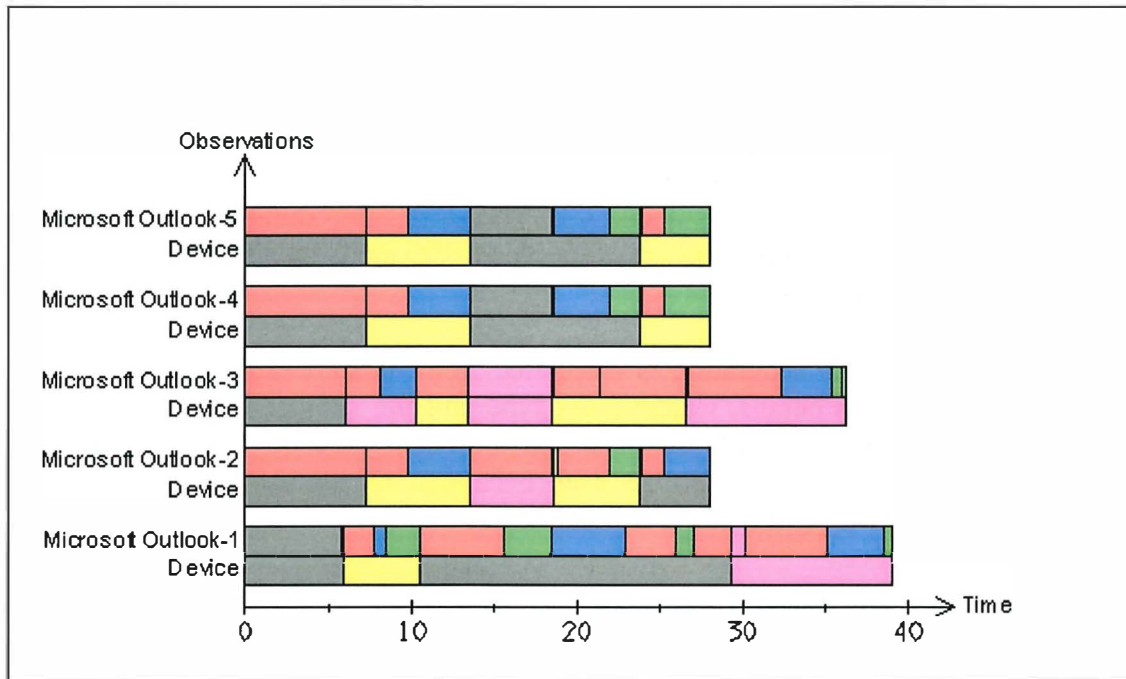


Figure 48: Temporal sequence of behaviors

On this figure, one can see five observations. For each of them, two bars are displayed: one is for the whole observation and is labelled with the observation's name, and the other concerns only the continuous behaviors performed by the device (and is labelled "Device"). Consequently, both bars have the same length.

Another point to underline is that for the correspondence between colours and behaviors, we had the choice between displaying a legend and making appear a popup menu when clicking on the behavior. It appeared to us that the second solution was the better one. Indeed, a legend making a potentially big number of colours appearing on the side of the chart would have overloaded it. We have therefore decided, when mouse is pressed on a behavior, to make appear data about it. This data is displayed only while the left button of the mouse is pressed, and it disappears if one starts dragging the mouse, since this action has the effect of translating the graph.

We will notice that dragging the mouse while maintaining the right button pressed allows one to zoom in and out, depending on whether the mouse is dragged towards the upper part of the screen or the bottom part of the screen.

Another remark to make is the fact that when the mouse cursor passes over a behavior, the cursor switches from the default one to the hand. And when the mouse leaves an area where clicking has the effect of displaying data, the default cursor is displayed again.

The data displayed when clicking on a particular behavior is composed of:

- The behavior's name,
- The behavior's subject
- The behavior's type
- The behavior's start time
- The behavior's duration

Figure 49 shows an example of such data being displayed:

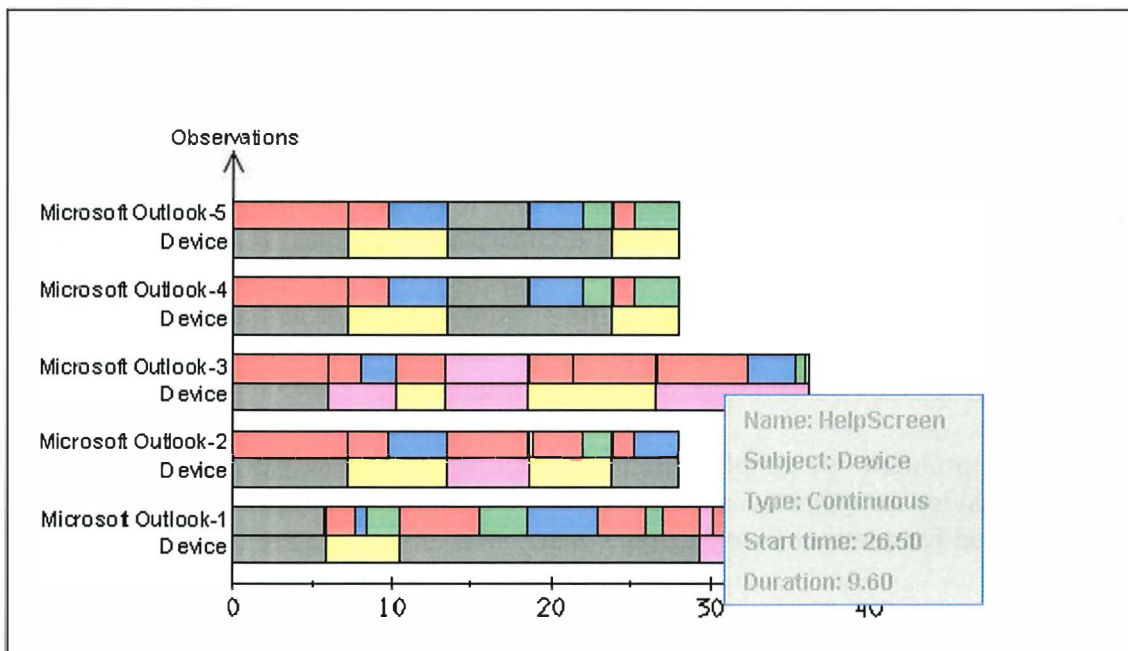


Figure 49: Display of the selected behavior's related data

Time/Frequency per behavior

In this second chart, we chose to visualise:

- The number of times each behavior has been observed in an particular observation
- The total time spent per behavior in a particular observation
- The number of times each behavior has been observed across observations
- The total time spent per behavior across observations

First, we refer to the observation selection for an appropriate selection of the observations. Selecting a single observation answers to the two first visualisation requirement while selecting all or some of them answers to the two last requirement.

The purpose of this graph is thus to visualise:

- The number of times each selected behavior has been observed in selected observations
- The total time spent per selected behavior in selected observations

Though the first measure refers to a time, while the second refers to a number, we decided to couple them because they are highly complementary. In fact, the main problem with the time spent per behavior is that it is only available for continuous behaviors. For example, it can be useful to compare the time spent reading or the time spent on a particular screen in several observations. But, it is impossible to compare the time spent clicking with the mouse or the time spent in an error as these behaviors are instantaneous. On the contrary, comparing the number of times an error has been observed in all selected observations can be very interesting.

To solve this problem, we simply present for each selected observation, both the time spent per behavior for continuous behaviors and the number of times each behavior has been performed for continuous and discrete behaviors.

To do that, as we are clearly in front of three dimensions, i.e. the observations, the behaviors and the time spent per behavior or the frequency to which the behaviors have been observed, we will use a standard three-dimensional chart: The column chart. A column chart typically displays discrete quantitative information by means of a series of geometric shapes such as rectangles, cylinders, ... Each column represents a data element, and a complete set of columns represents a data series.

The column chart is particularly efficient in our case as it is especially designed to look for relationships between multiple data series, i.e. behaviors in several observations.

Figure 50 shows the column chart of the time/frequency per behavior.

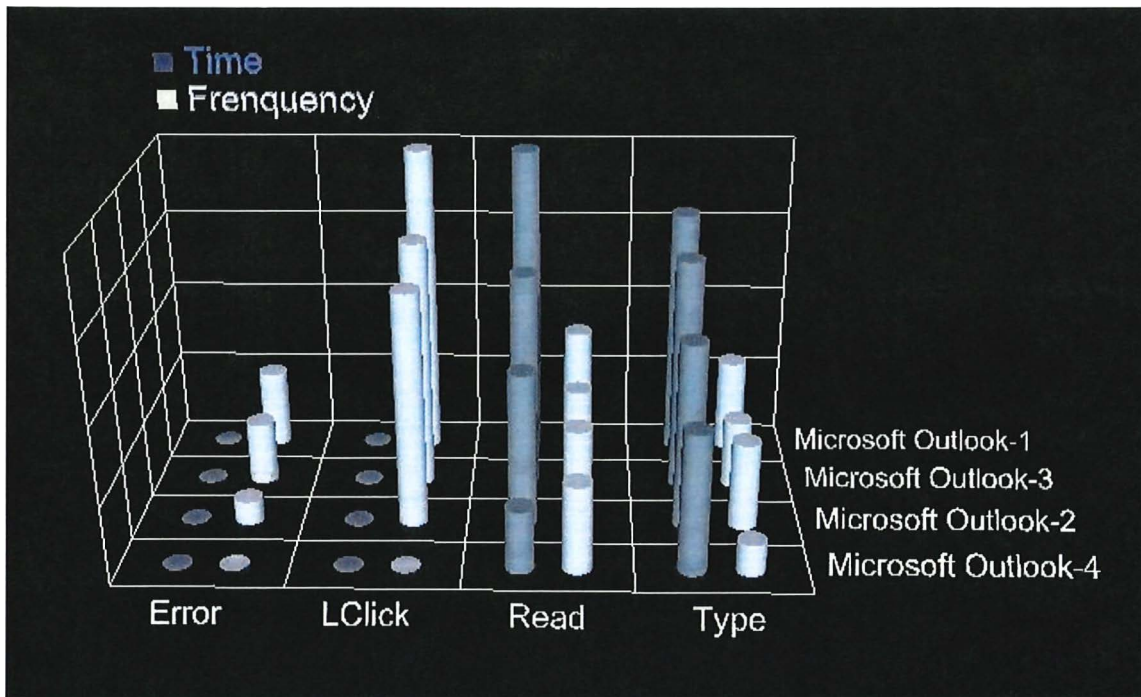


Figure 50: Time/Frequency per behavior

Behaviors are displayed on the X-axis, observations on the Z-axis and the time or frequency on the Y-axis. We also notice that no graduations are available on the Y-axis. The reason is that we did not want to overload the graph with a couple of scales as only the comparison from one observation to another is useful. However, the exact value can be obtained by clicking on the cylinder of interest.

Finally, to avoid confusion within this chart when it is displayed in a space limited area, we will limit the number of observations and behaviors simultaneously involved in this graph to five (the first five behaviors resulting from an alphabetical sort of the behavior names).

Sequential Analysis

In this graph, we present:

- The number of times each behavior has been followed or preceded by another in the completion of the task
- The number of times each behavior has been followed or preceded by another across observations

Like for the previous chart, we refer to the observation selection for an appropriate selection of the observations.

The aim of the Sequential Analysis is thus to visualise a squared matrix of the number of times each behavior has been followed by another across selected observations, i.e.:

$$A = [a_{ij}] \in \mathbb{R}^{n \times n}$$

where a_{ij} = number of times the behavior i has been followed by the behavior j

The main issue in the visualisation of this matrix is related to its content. More exactly, as some sequences are basically never observed, it contains a great number of 0. The sequential analysis should therefore provide a way to quickly focus on the non-zero values without according too much importance to zero values.

A second challenge we face is to allow the removal of all of parts of the elements contained in this matrix. For example, in the simple squared matrix shown on Figure 51, it is not relevant to know that an error message has led to a mouse click (the closure of the message dialog) or a reading period (the reading of this message) respectively 2 and 9 times across the selected observations. On the contrary, it is interesting to know that in most cases, errors occurred after a typing period (9 times) while using the mouse generally did not cause errors. This observation could lead to the introduction of combo boxes instead of typing fields or examples of the content of these fields.

	Error	LClick	Read	Type
Error	0	2	9	0
LClick	1	0	0	0
Read	0	4	0	2
Type	9	5	0	0

Figure 51: Example of matrix

As we have decided not to predefine behaviors (cf. Chapter 4), it is of the utmost importance to provide a way to visualise only parts of the initial squared matrix. We can furthermore notice that providing the ability to remove column of the matrix is much more useful than removing rows because we generally focus on the causes of a particular event we want to avoid (e.g. Error or Help screen) rather than on its consequences.

In order to meet these requirements, we introduce a 3D graph similar in the shape to the *Temporal Star* [Noirhomme, 2000]. The *Temporal Star* presents the advantage of displaying a set of *Simple stars* linked by a central vertical axis which represents the time, a *Simple Star* being a circular graph with a set of radial axes distributed equally around the 360 degrees of a circle.

In our case, The Simple Star represents the distribution of the behaviors following a particular behavior, while the central axis represents the different behaviors for which a distribution is available.

Figure 52 shows an example of the sequential analysis in the visualisation of the example matrix given above.

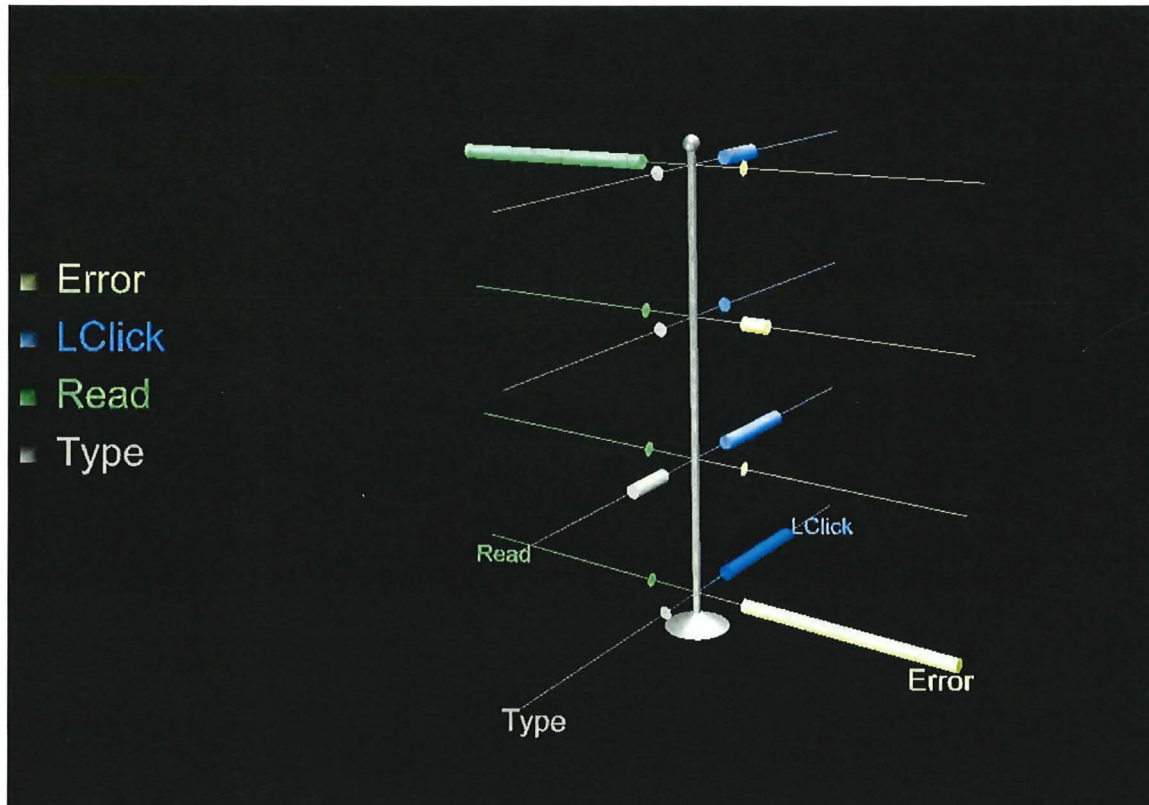


Figure 52: Sequential Analysis

The legend on the left side of the graph identifies each star downwards (e.g. the upper star gives the distribution of the behaviors following an error, the star below is related to the left click ...). Each star is composed of four axes, one for each behavior. The length of the cylinder on each axis represents the number of times the sequence has been observed. We notice that no graduations are available. The reason for that is that the exact distribution of the behaviors is not relevant as it is closely linked to the number of observations selected and the duration of these observations. Only the general trend in these sequences is important. However, the exact value can be obtained by clicking on a cylinder (cf. Figure 53).

This set of stars catches the attention on non-zero values and seems therefore appropriate for our needs. But a last issue is related to the removal of some columns of the matrix (i.e. axes of each star). This function is made possible by clicking in the legend on the behavior we want to remove from the stars.

Figure 53 shows the Sequential Analysis when focusing on the behaviors that have caused errors and reading periods. A shift in the position of a legend item points out that the behavior has been removed from each star. Of course, clicking a second time on an item puts back the behavior on the star.

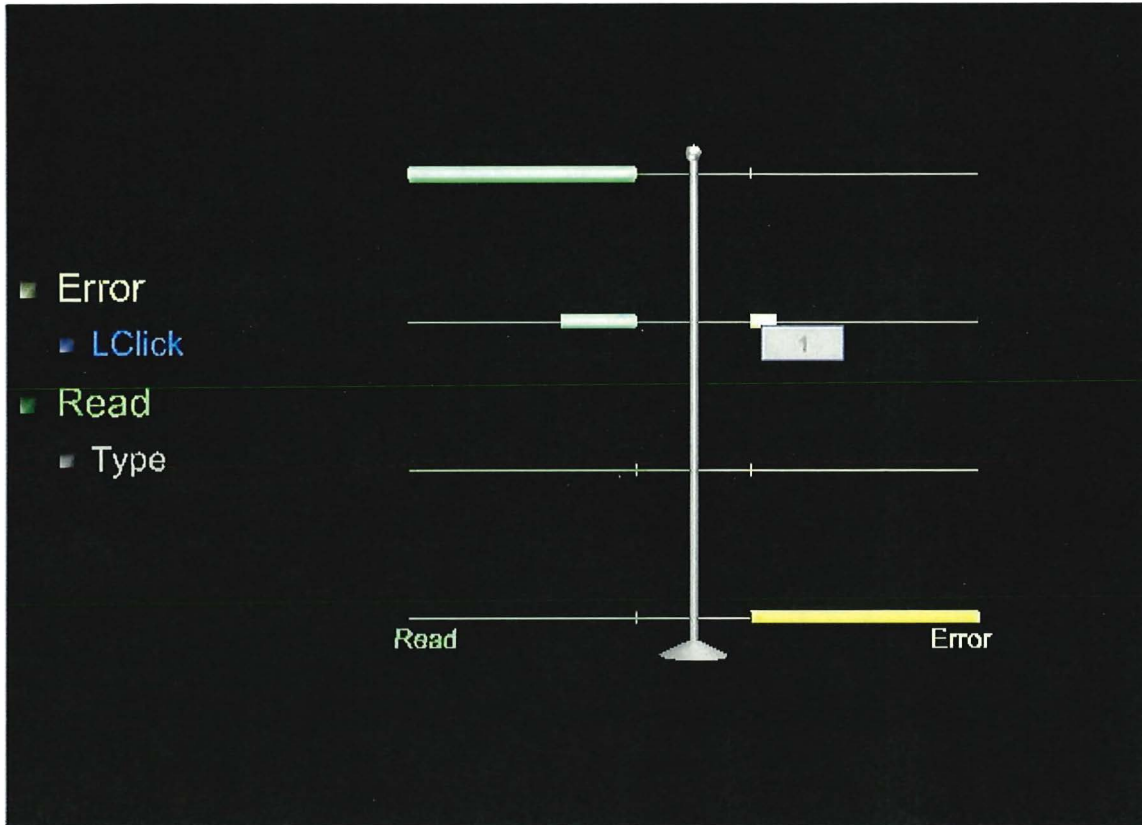


Figure 53: Sequential Analysis focusing on certain behaviors

To end this section, we precise that, given the limited available space for the display of this graph and to avoid the handling of huge matrices for each dynamic update of the graph, we will limit the number of simultaneously considered behaviors to five (the first five behaviors resulting from an alphabetical sort of the behavior names).

Optimal Sequence of Behaviors

This chart allows one to visualise the comparison between the sequence of behaviors followed by the user and an optimal sequence of behaviors.

Since we only interest ourselves in the sequence of the behaviors (i.e. the order in which they occur and follow each other), this chart does not make intervene time.

However, as mentioned in section 5.4.3, it is better for a chart to tend towards the horizontal. Therefore we have chosen to represent the sequences under the shape of horizontal bars. This is done as well to be consistent with the temporal sequence of behaviors. In section 5.4.3 is also mentioned the fact that the graphical perception research has found a preference for graphs nearing the so-called golden rectangle, in which the ratio of the longer axis to the shorter axis is 1.6 to 1. We therefore try to reach these proportions as closely as the disposition of the different views allowed us to.

As time does not intervene, the horizontal axis does not receive any scale value. Its label is "Sequence", in order to insist on the fact that the order of the behaviors is the object of this chart.

On the vertical axis, we make figure the observation names. Now we still have to consider the way of making possible the distinction between the different behaviors. Since the chart is designed to display different sequences of behaviors, it is impossible to make this dimension figure on an axis. When this data will be displayed, we will dispose of all the required data in order to perform an analysis. This thus answers the question of whether to insert 3D in the chart or not. That answer is very simple: 3D is not appropriate at all for this chart, since 2D is obviously fully sufficient to display all the required data. Therefore, the best way to introduce the behaviors is to represent each of them by a fraction of the bar it belongs to, i.e. by a rectangle. And then to attribute to each different behavior a distinct colour. This is done for a consistency reason, since the colours are the same as in the temporal sequence of behaviors. The main difference with the temporal sequence of behaviors is that, since time is not involved at all in this one, the behaviors will all be considered as having the same length, and will thus be represented by rectangles of equal size. Indeed, we insist again on the fact that this chart is designed to allow comparison between the observation sequences of behaviors and an optimal sequence, and thus that the focus is only put on the order of occurrence of those behaviors.

Another issue to address is the way to make the optimal sequence of behaviors appear. We had to choose between two possibilities. Either display that sequence once, for instance in the bottom of the graph, under all the observation bars. Or display it as many times as there are selected and displayed observations. The solution was chosen empirically: it indeed appeared to us that the graph was easier to use when displaying several times the optimal sequence, because, among others, it required less memory load from the user.

Therefore, this chart, just like the temporal sequence of behaviors, displays, for each observation, two different bars. Here, the first bar represents the optimal sequence of behaviors, and the second one the observation sequence.

Figure 54 shows the optimal sequence of behaviors chart:

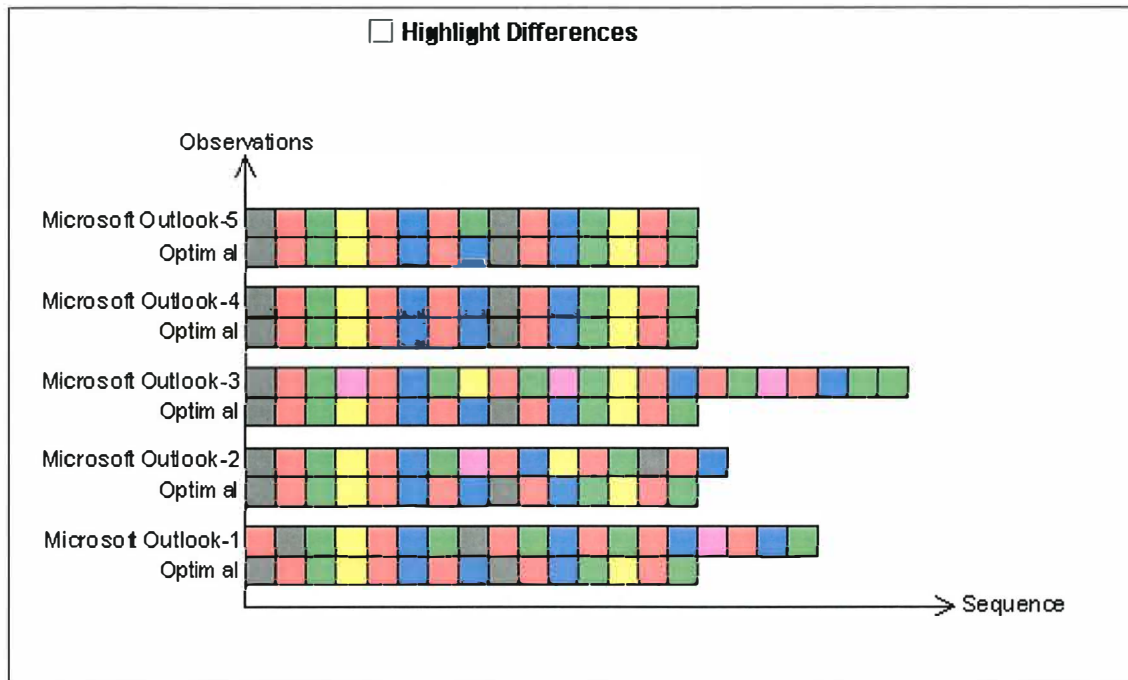


Figure 54: Optimal sequence of behaviors

On this graph, ten bars are displayed, that are grouped by two. Each group of two bars corresponds to one observation. The first one contains all the behaviors performed during the observation, and is labelled with the observation's name. The second one, labelled "Optimal" represents the optimal sequence of actions, and is the same whatever the observation.

In order to increase the visibility, we have added an option which allows one to highlight the differences between both the optimal sequence of behaviors and its related observation sequence of behaviors. This option is available through the "Highlight Differences" checkbox, and has the effect of comparing all the behaviors two by two: for instance, if the first behavior of the observation is the same as the first one of the optimal sequence, they will both be displayed in white. Otherwise they are displayed in red. This provides the possibility to locate immediately the places where both sequences diverge, and thus where are situated potential problems.

This can also show very easily whether the considered task can actually be completed through different ways, i.e. if different behaviors can actually lead to the same result, which is very often the case in most programs, in particular thanks to accelerators and other mnemonics.

See Figure 55 for an example of the above graph with highlighted differences:

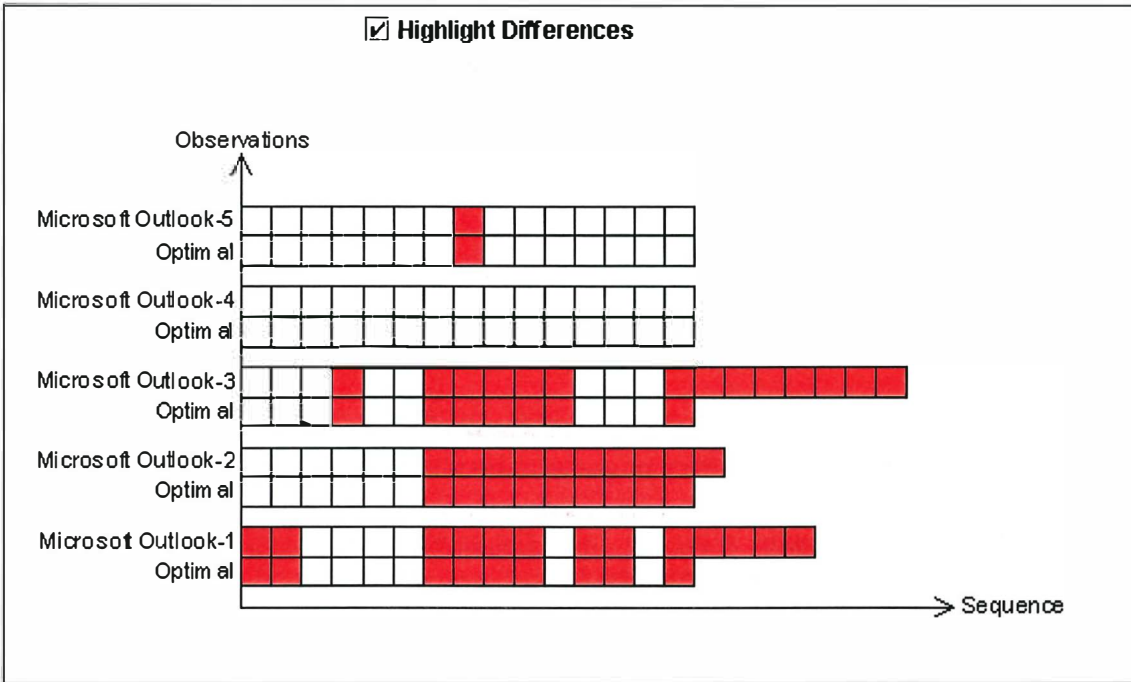


Figure 55: Optimal sequence of behaviors chart with differences highlighted

The popup menu, as well as the zoom and the translation, work exactly in the same way as in the temporal sequence of behaviors. However, the data displayed is different, in that only the selected behavior's name appears. From the behavior's name, by consulting the data displayed in the filtering panel, one can easily obtain other desired data. See Figure 56:

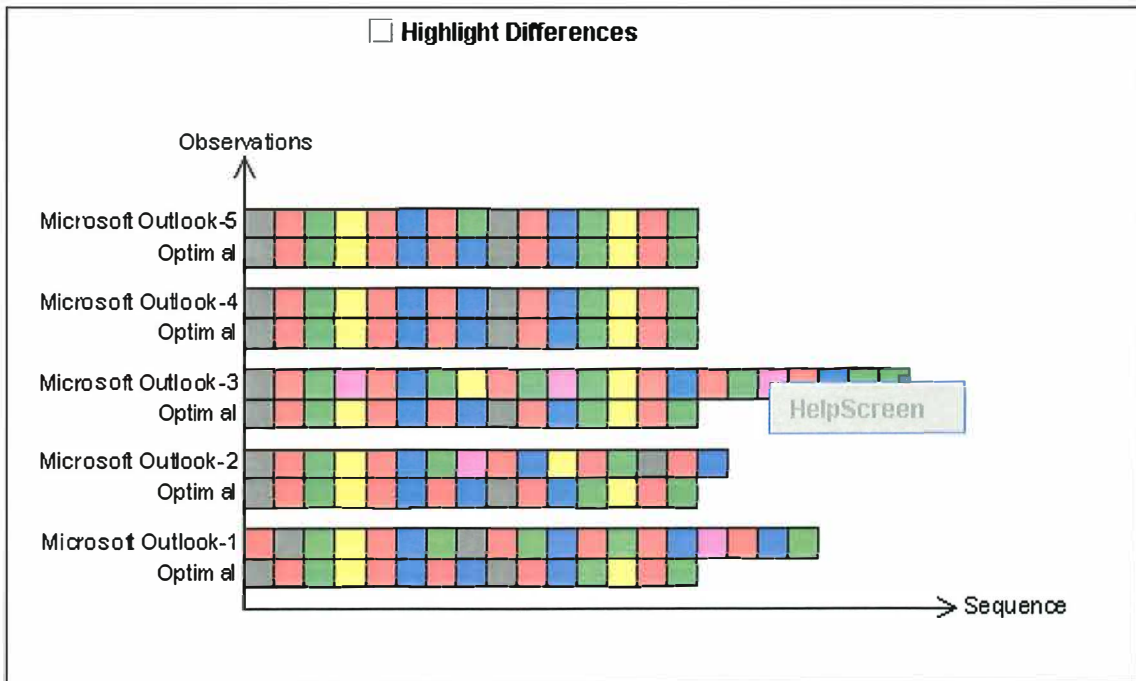


Figure 56: Display of the selected behavior's name

Statistics

The statistics provide simple information in a textual or tabular form on the completion of the task. This information is divided in two groups.

The first group is made of statistics collected in all selected observations together. It is basically question of average values that do not require any special visualisation techniques, i.e.:

- The completion rate or percentage of participants who completed the task correctly
- The mean task time or mean time taken to complete the task
- The completion rate efficiency or completion rate/mean task time

The second group displays in a tabular form the most basic information that can be collected on each observation individually, i.e.:

- If the user has completed the task successfully or not
- The total time taken to complete the task

Though they are quite simple, these statistics can be very helpful to get a quick idea on the global content and the general trend in the observations selected.

We also remind that further details on each observation can be obtained by double clicking on the observation of interest in the list of observations selected in the selection panel (cf. section 5.3.3).

Figure 57 shows an example of statistics.

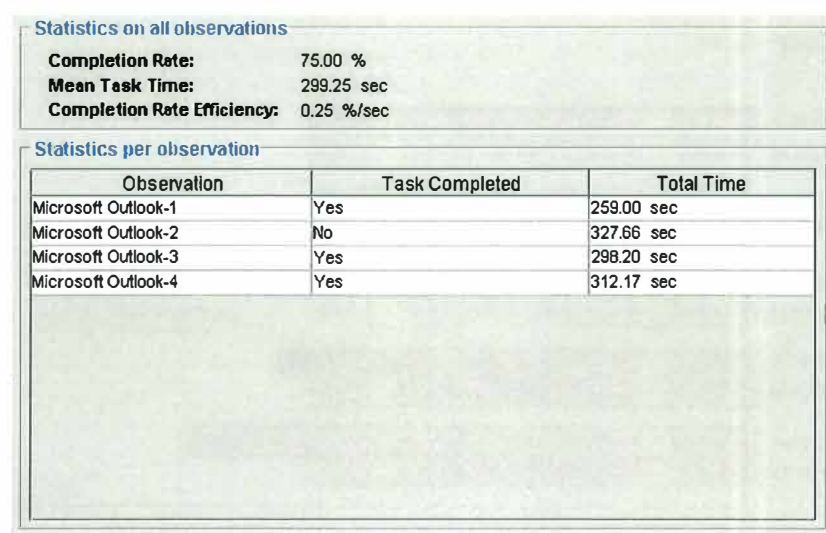
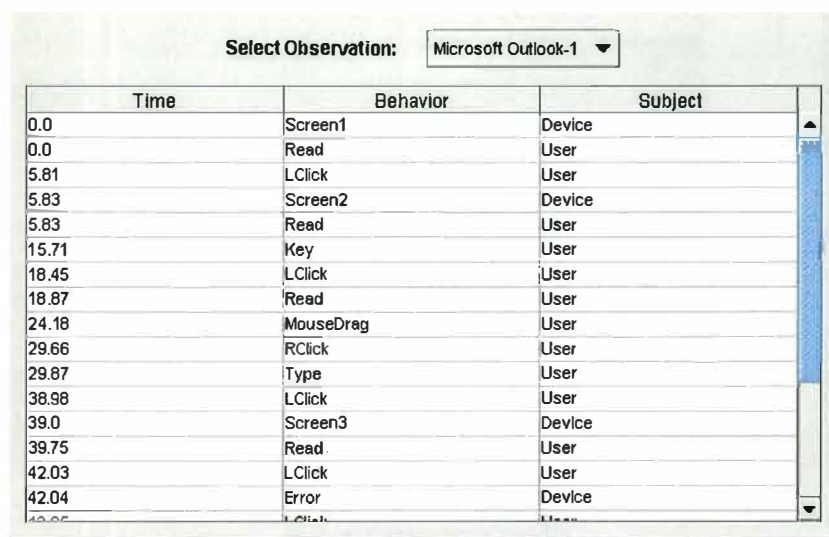


Figure 57: Statistics

Time Table

The Time Table gives in a tabular form, for each selected observation, the same information as the one contained in the observation part of an observation file. Though it does not provide any effective visualisation tool, we decided to keep it as it is essential for us to have an overview of what exactly contains each observation file before using other metaphors built on its content.

Because it is impossible to display this table simultaneously for all selected observations, we impose a second selection on them with the combo box shown on Figure 58. Further details on the behaviors contained in this table can be consulted in the filtering panel.



The screenshot shows a software interface with a dropdown menu labeled "Select Observation:" set to "Microsoft Outlook-1". Below it is a table with three columns: "Time", "Behavior", and "Subject". The table contains 17 rows of data.

Time	Behavior	Subject
0.0	Screen1	Device
0.0	Read	User
5.81	LClick	User
5.83	Screen2	Device
5.83	Read	User
15.71	Key	User
18.45	LClick	User
18.87	Read	User
24.18	MouseDown	User
29.66	RClick	User
29.87	Type	User
38.98	LClick	User
39.0	Screen3	Device
39.75	Read	User
42.03	LClick	User
42.04	Error	Device
42.05	LClick	User

Figure 58: Time Table

Video

Playing a video simply requires a video player. This player will be described in the following section about viewers.

5.4.5 Design of the viewers

The last step in the design of metaphors is the design of the viewers necessary to allow an effective handling of the data displayed.

Three kinds of viewers must be available in three distinct contexts:

- The handling of 2D graphs
- The handling of 3D graphs
- The play of a video

2D graphs handling

We identify three major tasks in handling 2D graphs. These are:

- The zoom
- The translation
- The reset task (reset the zoom and the translation)

Figure 59 shows the components used to perform these three tasks.

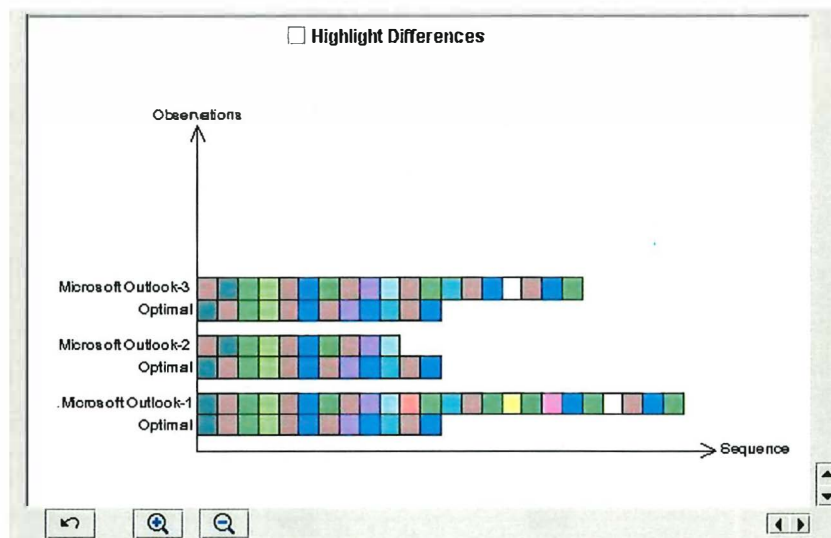


Figure 59: 2D Viewer

The zoom function is made possible through two standard "magnifying glasses" icons ("+" for zooming in and "-" for zooming out). The translation is provided with two couples of arrows. Horizontal arrows refer to the translation of the X-axis, while vertical arrows allow the translation of the Y-axis. The reset icon is situated on the left side of the zoom buttons. We further notice that both the translation and the zoom can be performed dragging the mouse, the first one with the left button pressed, the other with the right button pressed.

3D graphs handling

The three major tasks in handling a 3D graph in a virtual universe are the following:

- The zoom
- The rotation
- The reset task

The translation has not been retained to simplify the handling and as the 3D graphs described above do not really necessitate this task to be performed.

To be consistent, the zoom and reset tasks use the same buttons as for the 2D viewer. The rotation is possible using two scrollbars taking values between 0 and 360 (a complete rotation). The horizontal one allows the rotation of the graph about the Y-axis, the vertical one about the X-axis. We notice that the rotation about the Z-axis is not directly available as it is simply a combination of X and Y rotations.

Figure 60 shows the 3D viewer.

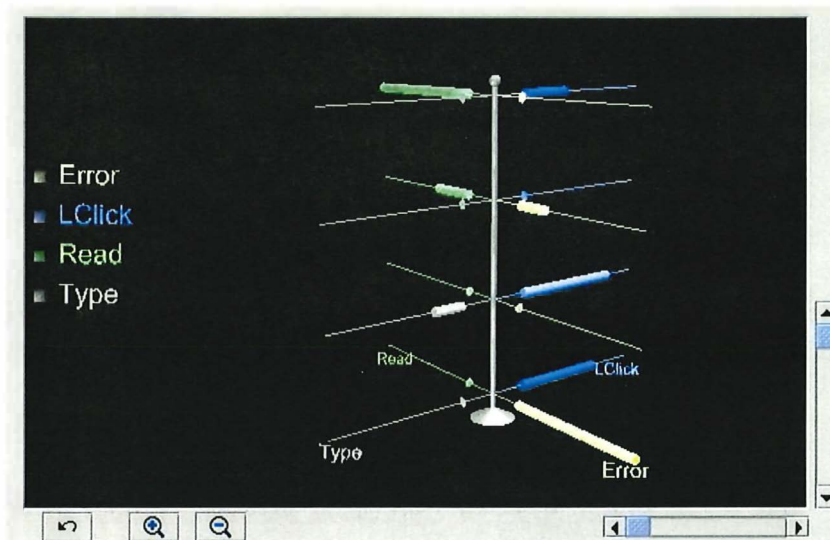


Figure 60: 3D Viewer

Like for the 2D graphs, the zoom and rotation can be performed dragging the mouse, respectively pressing the right and left button.

Video Player

The Video Player is shown on Figure 61. It uses standard buttons and allows the following actions:

- Open a media file ("Open" icon)
- Stop ("Stop" icon)
- Play ("Play" icon)
- Pause ("Pause" icon)
- Change the position on the video (Slider below the video screen)
- Change the volume (Slider on the left side of the video screen)

Furthermore, the total time and current time (or time "elapsed") are displayed on the right side of the video screen.



Figure 61: Video Player

The "Open" icon opens a secondary window containing all the media files referenced in the configuration and observation files (cf. Figure 62). When a file is open, its title is added to the title of the view.

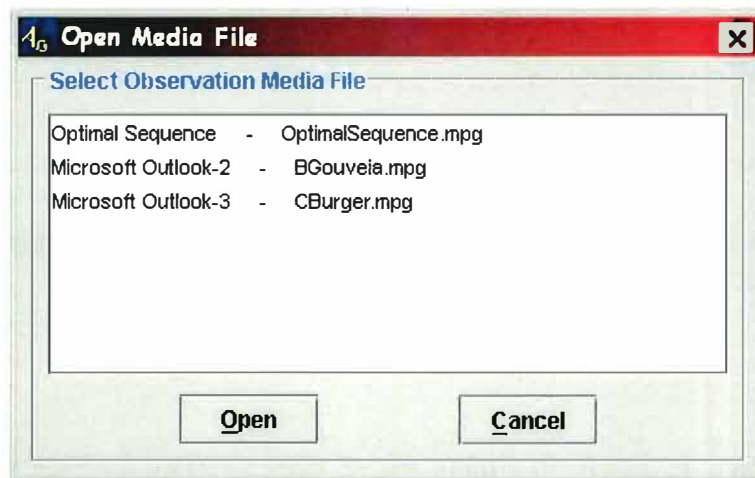


Figure 62: "Open media file" window

5.4.6 Interaction between views

The Spotfire analysis (cf. section 5.3.2) has outlined the interest of a brushing interaction. A brushing interaction is a type of link between multiple views, in which items highlighted in one view are automatically highlighted in other views by the system. However, though this means of interaction seems to be very useful, we will show in the following that this type of interaction is not suitable for the analysis of usability data.

Each of the metaphors described above provides an investigation tool designed in a particular purpose. Although the items displayed in different views present similar characteristics, their semantic is therefore in most cases different. The main reason for that is that, contrary to what happens in Spotfire, all these metaphors have been designed to avoid redundancy in order to satisfy to the principle of parsimony. Because of this dissimilarity, we first think that the use of such an interaction could introduce confusion by linking items for which there is no real comparison possible.

An other direct consequence of this problem is that the result of the interaction process can be different depending on the view in which an item has been selected. For example, highlighting a particular action in the sequence of behaviors should result in the highlighting of the cylinders related to this behavior in the 3D graph showing both the number of times this behavior has been observed in the corresponding observation and the time spent performing this behavior in the observation. On the contrary, highlighting a cylinder of this last graph should result in the highlighting of all actions involving this behavior. The reason for that problem is that these apparently related items have very different meanings. Items in the sequence of behaviors refer to actions while items in the other graph refer to numbers or times spent performing a particular behavior.

A solution for this problem could be to highlight all items involving the same behavior in all views. But in this case, we think that the result of the interaction would be much more confusing than useful. Furthermore, the main goal of the interaction would not be reached in certain cases where a brushing interaction is suitable. For example, as the sequence of behaviors and the time table show the same thing, highlighting an item in the sequence should highlight the corresponding item in the table and not all identical behaviors in the sequence and the table.

These examples show that a brushing interaction is not possible in our system without introducing complexity due to a fuzzy logic. Furthermore, the difference in the semantic of the items contained in the views let us think that we will not be able to connect the data in different views in a simple and useful way.

5.4.7 Conclusion

In the conclusion of this chapter, we present a summary of how we have translated the functional requirements developed in Chapter 4 into concrete components of our system. For each task of the generalised task model (section 4.5.4), we refer to the corresponding section and/or component that cover the subject in order to check that none of them have been forgotten.

Task	Reference
1. Open project containing observation files in a specified format	- File format for the data, p115-122
2. See information about the observations	- The consultation of information about observations, p144-145
3. Select specific observations	- The selection of the observations, p140
4. Filter data	- The filtering of the data, p145-146
5. Visualise data	- The visualisation panel, p147-148 - Identification and design of the visualisation metaphors, p159-173
6. Zoom data (in/out)	- Cf. tasks 2 and 3 - Design of the viewers, p173-175
7. Undo/Redo actions performed	- Identification and design of the visualisation metaphors, p159-173
8. View details on demand	- The menus, p149-150 - The toolbar, p150-151
9. Save views to a file	- The menus, p149-150 - The toolbar, p150-151
10. Print views	- The menus, p149-150 - The toolbar, p150-151

The sections referenced in this table describe in detail the design choices we made. We can further notice by examining these sections, that all of them have been carefully justified with regard to the corresponding functional requirements and, if possible, using guidelines. A last point related to the design of the system that has not yet been addressed is the evaluation and validation of these choices. This evaluation is discussed in Chapter 7.

Chapter 6. Implementation

6.1 Introduction

The purpose of this chapter is to discuss the implementation of the application. We first explain the reasons that drove us to use the Java technology. Then, we introduce the main elements helping to understand and maintain the code. In this second part, we start with a diagram presenting the hierarchy of all interface-oriented components used with a reference to the corresponding Java classes, and finally we give the programming conventions used within these classes.

6.2 Implementation tool

In this section, we motivate our choice for a system fully implemented in Java. Three points justify the use of Java within the framework of this thesis:

We first refer to the design constraints in section 4.5.4. These constraints are:

- The system must be platform independent
- The system must be window-based

As platform independence is required, Java is suitable to fulfil this requirement. Of course, Java is not the only system that provides such platform independence, but it is the first to achieve widespread popularity among commercial developers. According to [Tyma, 1998]:

"Platform independence in Java really takes two forms. The popular notion is that we write our code, compile it and then never worry about having to port it to new machines (write once, run everywhere). The more overlooked side of platform independence is Java's rather fantastic abstraction of many programming paradigms".

A second point that confirms our first choice is given in El ansari and Vanbrabant thesis [El ansari and Vanbrabant, 2001]. This study is a validation of Wesson's arguments that Java technology is mature enough to provide the tools for data visualisation. As our thesis is a continuation of both Darville and Van Espen work [Darville and Van Espen, 2000] and El ansari and Vanbrabant work, using Java seems appropriate to confirm their research by developing a system fully based on this technology.

However, a last issue is related to the use of three-dimensional contents. Both [Darville and Van Espen, 2000] and [El ansari and Vanbrabant, 2001] make use of Java in combination with the VRML technology instead of the Java3D API (Application Programming Interface) for the design of three-dimensional contents.

As a reminder, VRML is an acronym for the Virtual Reality Modelling Language. It is a web-centric technology which allows the user to describe interactive three-dimensional worlds. We further notice that VRML is not a programming language, but a descriptive language.

On the contrary, Java is a programming language that typically defines an Application Programming Interface (API) and involves code compilation, libraries linking and a restrictive syntax. The Java3D API is used to design three-dimensional graphics applications or applets. It is a standard extension to the Java2 Platform that provides a collection of high-level constructs in order to create and manipulate 3D geometry as well as structures to render this geometry.

The main problem of the VRML technology is that it is only usable with a VRML browser that interprets, renders the 3D contents and usually provides a navigation tool to move within the virtual world. VRML browsers are typically installed as plug-ins within traditional browsers such as Netscape Navigator and Internet Explorer. VRML objects can thus be linked to the Java technology through a Java applet, but using VRML directly in an application is impossible except if using Java3D loaders to parse and translate VRML files and generate the appropriate Java3D objects and Java code to support the files contents.

As the second design constraint imposes to develop an application and not an applet, VRML is not suitable at all for the implementation of our system. Furthermore, a programming language is much more suitable, since it is more powerful than a descriptive language and above all more appropriate for the dynamic and interactive visualisation tool we want to develop. The question that immediately emerges from this discussion is the following: Is Java3D suitable for data visualisation?

The answer to this question can be found in Darville and Van Espen thesis "Study of the Virtual Reality Technology used in the Visualisation of Business Information" [Darville and Van Espen, 2000]. In their research on the technological requirements needed to visualise Business data, they compared VRML and Java3D on the basis of the following technological requirements that the ideal technology should fulfil:

1. Dynamic chart building and behavior programming
2. Display of three-dimensional contents
3. Navigation and interactivity
4. Availability of basic elements found in today's 3D applications
5. Ease of development
6. Usable Graphical User Interface
7. Performance
8. Universality (File interchange format, cross-platform and network-centric application)

Their conclusion is given below:

"Java3D meets our requirements list. As Java3D belongs to the JavaMedia Suite, it inherits from all its advantages. It is able to create sharp visual effects and many behaviours as well as improve automatically the rendering in many ways. Therefore, Java3D is also a perfectly valid development platform for Business Information Visualisation purposes.

However, this platform demands more development time than in VRML. Despite a whole bunch of features to make the Java3D API more accessible (scene graph, automatic rendering, and so on), Java3D keeps a very restricting syntax. Furthermore we should program our own navigation tool since powerful browsers, like in VRML, do not exist in Java3D."

According to them, Java3D is a perfectly valid development platform. It only presents the disadvantages of demanding more development time (that makes sense since VRML is a descriptive language) and of not providing any navigation tool which is not unacceptable.

However, another disadvantage of Java 3D was, according to them, the instability of its API. All we can say about it is that after performing numerous tests, we did not encounter serious problems. Probably the last releases of Java 3D have been largely improved.

There was thus no major objection for using Java in 3D data visualisation and for the development of our system.

To end this section, here are the Java tools we used in order to allow the development of all the components defined in Chapter 5:

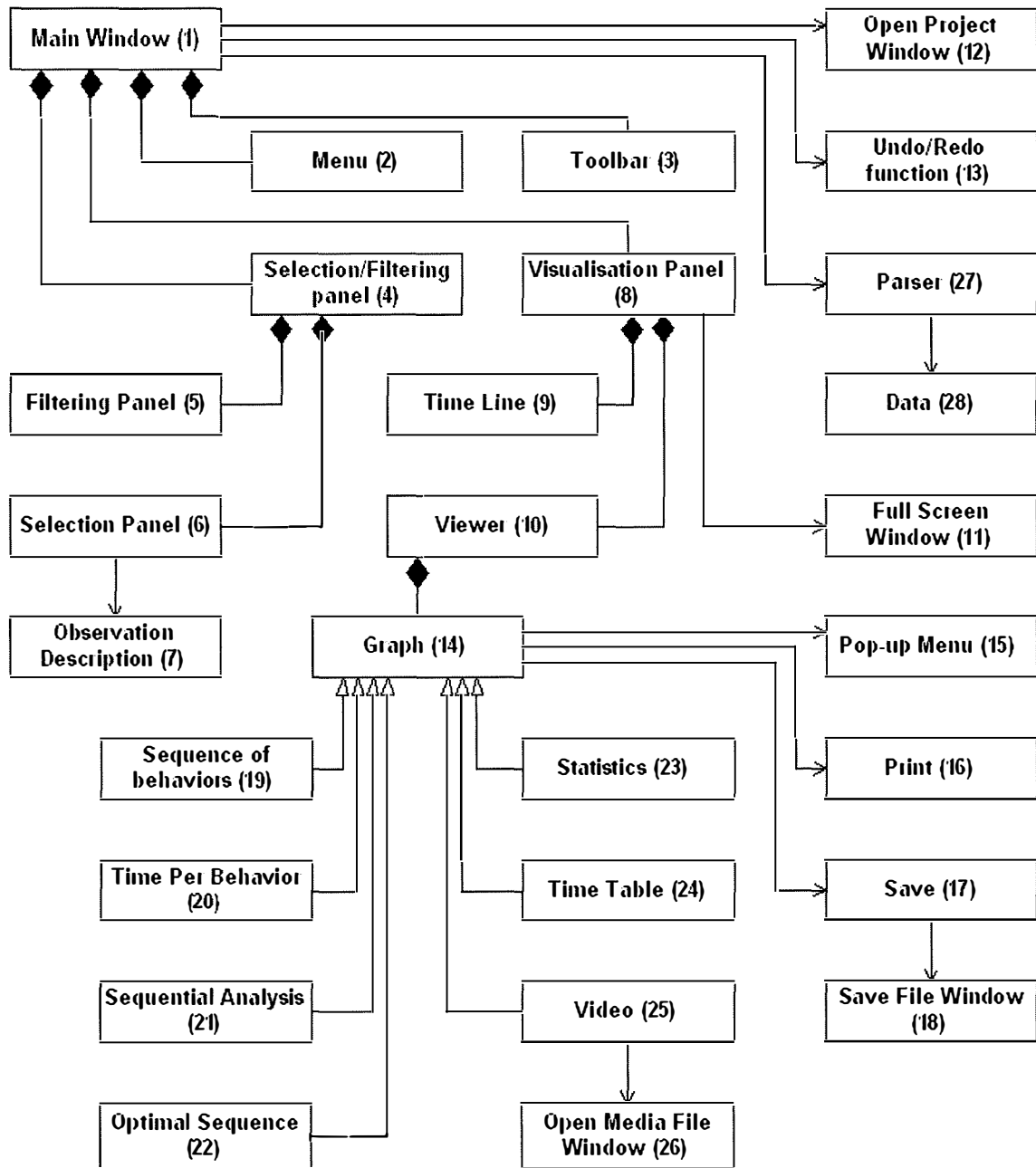
- Java 2 SDK, Standard Edition, v1.3.1_02 for the Graphical User Interface development
- Java 3D 1.2.1_03 SDK (DirectX version) for the design of three-dimensional contents in a virtual universe
- Java Media Framework 2.1.1a for the Video Player

6.3 Implementation

6.3.1 Introduction

The purpose of this section is to give the main elements helping to understand the general structure of the Java classes implemented. To do that, we present an interface-oriented class diagram of all concrete components within the application. For each of them, we refer to the corresponding Java classes that implement these components. Finally, we precise the main programming conventions used within these classes in order to improve the readability of the application, allowing future workers on this project to understand the code more quickly and thoroughly. These conventions mostly reflect the Java language coding standards presented in "*The Java Language Specification*" from Sun Microsystems [Java, 2000]

6.3.2 Class Diagram



Notations

- ◆ Is contained in
- > Uses
- ▷ Is A

Figure 63: Structure of the application

Legend

Package "Interface"

- (1) AlgoaGraphics.java, PanneauPrincipal.java
- (2) Menu.java
- (3) BarreOutils.java
- (4) PanneauSelections.java
- (5) PanneauFiltrageDonnees.java, CheckableItem.java, CheckCellRenderer.java, ModeleTableComportements.java
- (6) PanneauSelectionObservations.java, SpinBox.java, IntField.java
- (7) FenetreDescriptionObservation.java, FenetreSuiteDescription.java
- (8) PanneauVues.java, PanneauVueUnique.java, PanneauDeuxVues.java, PanneauTroisVues.java
- (9) PanneauLigneTemps.java, DoubleSlider.java, DoubleSliderUI.java, MetalDoubleSliderUI.java
- (10) PanneauInterneVues.java
- (11) FenetreVue.java
- (12) BoiteOuvertureProjet.java, SelecteurFichier.java, VueFichier.java
- (13) UndoRedo.java

Package "Graphiques"

- (14) Graphique.java
- (15) MenuContextuel.java
- (16) Impression.java, Impression3D.java
- (17) Sauvegarde.java, Sauvegarde3D.java, FichierBmp.java
- (18) BoiteSauvegardeImage.java
- (19) Sequence.java
- (20) TimePerBehavior.java
- (21) SequentialAnalysis.java
- (22) OptimalSequence.java
- (23) Statistiques.java, ModeleTri.java
- (24) TimeTable.java, ModeleTable.java
- (25) Video.java, MetalCurseurSliderUI.java, MetalVolumeSliderUI.java
- (26) BoiteOuvertureMedia.java

Package "Donnees"

- (27) Parseur.java
- (28) Data.java, EntreeObservation.java, InfoObservation.java, EntreeBehavior.java

6.3.3 Programming conventions

Java Source Files

Each Java source file contains a single public or package level (no access modifier) class or interface. When private classes and interfaces are associated with a public class, we put them in the same source file as the public class. The public class is the first class or interface in the file.

Java source files have the following ordering:

- Package and Import statements
- Class and interface declarations

Package and Import Statements

The first line is a package statement. After that, import statements can follow. For example:

```
package Graphique;  
  
import Donnees.*;
```

Class and Interface Declarations

The following table describes the parts of a class or interface declaration, in the order that they appear.

Part of Class/Interface Declaration	Notes
1. Class/interface documentation comments	See “Documentation Comments”
2. Class or interface statement	
3. Instance variables	First <i>public</i> , then <i>protected</i> , then package level (no access modifier), and then <i>private</i> .
4. Constructors	
5. Methods	First <i>public</i> , then <i>protected</i> , then package level (no access modifier), and then <i>private</i> .

Indentation

Four spaces are used as the unit of indentation. Tabs are set exactly every 8 spaces.

Line Length

No lines are longer than 120.

Wrapping Lines

When an expression does not fit on a single line, it is broken according to these general principles:

- Break after a comma
- Break before an operator
- Align the new line with the beginning of the expression at the same level on the previous line.

Comments

There are two kinds of comments: implementation comments (i.e. comments about the particular implementation) and documentation comments (i.e. comments that describe the specification of the code).

Implementation comments are delimited by `/*...*/`, and `//` and are mostly written in French for our own facility.

Documentation comments (known as “doc comments”) are Java-only, and are delimited by `/**...*/`. Doc comments are provided with the application in HTML files extracted using the javadoc tool.

Documentation Comments

Doc comments describe Java classes, interfaces, constructors, methods, and sometimes fields. Each doc comment is set inside the comment delimiters `/**...*/`, with one comment per class, interface, or member. This comment appears just before the declaration as follows:

```
/**
 * The Example class provides...
 */
public class Example { ...
```


Declarations

Number Per Line

One declaration per line is made.

Initialisation

Local variables are initialised where they are declared. The only case when they are not is when the initial value depends on some computation occurring first.

Placement

Declarations are put only at the beginning of blocks. (A block is any code surrounded by curly braces “{” and “}”.)

Class and Interface Declarations

When coding classes and interfaces, the following formatting rules are followed:

- No space between a method name and the parenthesis “(“ starting its parameter list
- Opening and Closing brace start a line by itself and are indented to the beginning of the declaration, except when it is a null statement, in which case the “)” appears immediately after the “{“

Statements

Simple Statements

Each line contains at most one statement.

Compound Statements

Compound statements are statements that contain lists of statements enclosed in braces, “{ statements }”.

- The enclosed statements is indented one more level than the compound statement.
- The opening brace and closing brace start a line and are indented to the beginning of the compound statement.

Naming Conventions

Naming conventions make programs more understandable by making them easier to read. Again, for our own facility, most names are written in French. The following table gives the main naming conventions:

Identifier Type	Rules for Naming
Packages	Class names are nouns with the first letter capitalised.
Classes	Class names are nouns, in mixed case with the first letter of each internal word capitalised.
Interfaces	Interface names are capitalised like class names.
Methods	Methods are verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalised.
Variables	Variables are in mixed case with a lower-case first letter. Internal words start with capital letters.
Constants	Constants are uppercase.

Chapter 7. Evaluation of the System

7.1 Introduction

In this last chapter, we carry out a global evaluation of the system we developed. The conclusion of Chapter 5 has already shown that the system fulfils all functional requirements defined in Chapter 4. In this chapter, we particularly focus on the usability of the system.

Though it would have been better to have this evaluation performed by the expert users on usability testing for which this application has been designed, we will perform it by ourselves, and this, for the following reasons:

First, we do not have at our disposal the same usability environment as the one we had at UPE. The University of Namur where we currently complete our last year study, is very little specialised in the field of usability testing. It is thus very difficult for us to benefit from the help and skill of other students or searchers to perform the evaluation without requiring from them an extra learning about the basic concepts of usability. It is furthermore impossible to lead a usability test from scratch as we do not have other alternatives than generating the log files manually.

And second, since all the functions available within the application have been completed only a few times before the deadline for this thesis, we did not have the necessary time to contact experts outside the University.

The different methods available to have usability-related aspects of a user interface examined by usability specialists are called usability inspection (cf. Section 2.2.3). All of them obviously match our needs except the Pluralistic walkthrough which involves too many testers. However, some of them are too time-consuming and ask too much preparation within the framework of this thesis, more especially as the evaluation of our system is only a sub-goal of this thesis. This is the case of the Guideline review and the Cognitive walkthrough. On the contrary, other methods only focus on a particular aspect of the evaluation and are therefore not suitable for the global evaluation we want to perform. We can mention the Standards inspection, the Consistency inspection and the Feature inspection.

For all these reasons, we decided to perform the usability evaluation using a simple but complete method for finding the usability problems of a user interface. This technique, already exposed in Chapter 2 and developed in Chapter 3 when analysing the usability of both WebTrends Log Analyser and Noldus Observer Video Pro, is the heuristic evaluation.

As a reminder, a heuristic evaluation is a method for structuring the critique of a system using a set of relatively simple and general heuristics that can be used to generate ideas while critiquing the system [Shneiderman, 1998]. These heuristics are:

- Visibility of system status
- Match between system and the real world
- User control and freedom
- Consistency and Standards
- Error prevention
- Recognition rather than recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help user recognise, diagnose and recover from errors
- Help and documentation

This technique seems very appropriate in our case, as it only involves simple usability heuristics, without requiring from us to critique the foundations of our system, for which we are obviously badly placed.

7.2 Heuristic evaluation

7.2.1 Introduction

In order to perform a heuristic evaluation, we have to determine for each major task of the generalised task model (cf. section 4.5.5), if it satisfies the ten heuristics defined above. The three tasks we will focus on are the three main tasks in the process of visualising usability data, i.e.:

- The selection of observations
- The data filtering
- The data visualisation

7.2.2 The selection of observations

The selection of observations can be performed directly or on the basis of independent variables related to the users involved in the observations, i.e. the gender, the age, the language and the expertise. This task is made possible through the "Select Observations" panel on the left side of the selection and filtering panel. The heuristic evaluation is given below:

Heuristics	Evaluation	
	<i>Positive</i>	<i>Negative</i>
Visibility of system status	- List of selected observations always available	\
Match between system and the real world	- Appropriate and logical means of selection	- Terms "independent variables" not explicit
User control and freedom	- Undo and Redo actions available for every single action within the observation selection	\

Consistency and standards	<ul style="list-style-type: none"> - Standard means of selection (radio buttons, checkboxes, spinboxes, system of two lists) - Standard icons 	<ul style="list-style-type: none"> - Double clicking on an observation to get details is not really intuitive though a tooltip text indicates this function is available
Error prevention	<ul style="list-style-type: none"> - No errors possible 	\
Recognition rather than recall	<ul style="list-style-type: none"> - All options directly related to the observation selection are visible - No need to recall anything since all pertinent data is displayed 	\
Flexibility and efficiency of use	\	<ul style="list-style-type: none"> - No accelerators (not really necessary) except for undo and redo functions
Aesthetic and minimalist design	<ul style="list-style-type: none"> - No extra information 	<ul style="list-style-type: none"> - Relative overload of the selection panel
Help user recognise, diagnose and recover from errors	<ul style="list-style-type: none"> - No possible error 	\
Help and documentation	<ul style="list-style-type: none"> - Documentation available (cf. Chapter 5) 	<ul style="list-style-type: none"> - No help function

7.2.3 The data filtering

The filtering of data includes:

- The selection of the subjects (user and computer)
- The selection of the behaviors performed by these subjects
- The selection of the time period

This task involves mainly the "Filter data" panel on the right side of the selection and filtering panel and the time line, both defined in Chapter 5. The evaluation for this task is given below:

Heuristics	Evaluation	
	<i>Positive</i>	<i>Negative</i>
Visibility of system status	- Current selection always visible	\
Match between system and the real world	- Appropriate labels - Logical means of filtering - Selection of the time period very intuitive	\
User control and freedom	- Undo and Redo actions available for every single action within the data filtering	\
Consistency and standards	- Standard means of filtering (radio buttons, checkboxes, sliders) - Standard icons	\
Error prevention	- No possible error	\

<p>Recognition rather than recall</p>	<ul style="list-style-type: none"> - Visibility of the most useful selection options - No need to recall anything 	<ul style="list-style-type: none"> - The possibility to right click on a table column's header in order to sort that column's content is not explicitly visible - Neither is the possibility to double click on the description of a behavior to get the complete description when it is too long
<p>Flexibility and efficiency of use</p>	<p>\</p>	<ul style="list-style-type: none"> - No accelerators (not really necessary) except for undo and redo functions
<p>Aesthetic and minimalist design</p>	<ul style="list-style-type: none"> - No extra information - Minimalist design and good visibility 	<p>\</p>
<p>Help user recognise, diagnose and recover from errors</p>	<ul style="list-style-type: none"> - No possible error 	<p>\</p>
<p>Help and documentation</p>	<ul style="list-style-type: none"> - Documentation available (cf. Chapter 5) 	<ul style="list-style-type: none"> - No help function

7.2.4 The data visualisation

The visualisation of the selected observations and behaviors is provided with the multiple views displayed in the visualisation panel. The facility with which one can switch from a certain number of views to another and from one chart or table to another one as well as the charts handling are evaluated in the table below:

Heuristics	Evaluation	
	<i>Positive</i>	<i>Negative</i>
Visibility of system status	<ul style="list-style-type: none"> - Chart/table names and view names always visible - Current organisation of the views displayed available in the toolbar 	\
Match between system and the real world	\	<ul style="list-style-type: none"> - Chart names not always explicit
User control and freedom	<ul style="list-style-type: none"> - Undo and Redo actions available when changing the number of views displayed and switching from one chart or table to another 	<ul style="list-style-type: none"> - Undo and redo actions not available for the handling of the charts
Consistency and standards	<ul style="list-style-type: none"> - Standard icons 	\
Error prevention	<ul style="list-style-type: none"> - No possible error 	\
Recognition rather than recall	\	<ul style="list-style-type: none"> - Options to change the number of views and to switch charts hidden in the menu bar and the pop-up menu

Flexibility and efficiency of use	<ul style="list-style-type: none"> - Icons in the toolbar to change the number of views - Pop-up menu to switch from one chart or table to another 	\
Aesthetic and minimalist design	<ul style="list-style-type: none"> - No extra information - Minimalist design 	\
Help user recognise, diagnose and recover from errors	<ul style="list-style-type: none"> - No possible error 	\
Help and documentation	<ul style="list-style-type: none"> - Documentation available (cf. Chapter 5) 	<ul style="list-style-type: none"> - No help function

7.3 Conclusion

This quick evaluation of the system has not shown any fundamental usability problems except a clear need for a help unit. But these favourable results are partially logical, first because we kept in mind the ten heuristics listed above during the whole interface design process, and second because we based our design on reliable guidelines for the design of interfaces.

Nevertheless, as it is said in the introduction of this chapter, no real evaluation of the aptitude of this system to meet the needs of experts in usability testing has been performed. Working in collaboration with experts during the last months of this project would have probably changed the way we dealt with some fundamental issues such as the design of the visualisation metaphors.

However, it seems reasonable for us to think that this system is more than acceptable, and we are sure that, after a deep evaluation based on experts recommendations, future workers on this project will put to good use what has been done until now.

Chapter 8. Conclusion

8.1 Achievements

The main goal of this thesis has been achieved through the development of a fully functional visualisation system. This system, designed for expert users in usability testing, can be used to interactively visualise quantitative usability data resulting from tests performed in a usability laboratory.

As a sub-goal of this project, an analysis of competitive tools that allow one to collect and visualise usability data had to be performed. As another sub-goal, an investigation had to be conducted to determine the semantics of typical usability data. A further sub-goal was to design the interface and to determine the most suitable metaphor(s) to use to visualise this data. A final sub-goal was to determine the most appropriate implementation tool(s) to use.

After a short introduction to formal usability testing and usability laboratories, we presented, in Chapter 2, a complete literature review of the usability field. In this review, we first introduced the most commonly accepted definitions of the term "usability". Then, we developed the different stages of the usability engineering lifecycle by which we have been largely inspired in the development of our system. Next, we deepened the different techniques available to perform a usability evaluation, and more particularly the testing methods available for a test in a usability laboratory. And finally, we gave some general recommendations on the way to analyse the data collected within the testing methods defined.

Chapter 3 was divided in two parts. In the first part, we dealt with a fundamental step of the usability engineering process: the competitive analysis. To do that, we first analysed heuristically two complementary existing systems: the WebTrends Log Analyser and the Noldus Observer Video Pro. Then, we analysed in detail the visualisation techniques available within these two systems. This second analysis, much more interesting in its results than the first one, led to a set of useful recommendations. Among them, a need for more interactivity in the charts and table produced, including the possibility to change the view, to zoom in and out and to get further details on items displayed.

In the second part, we addressed the first sub-goal of this thesis mentioned in the problem statement: the investigation into the semantics of typical usability data. After a restriction of the definition of typical usability data to typical quantitative usability data, we gave a list of common characteristics that should be present in it. Then, we presented a general schema of the necessary data to fulfil this list of requirements in order to give an overview of the data we considered as minimum to allow a quantitative usability data analysis with our system.

Once the scope of this thesis has been defined, we dedicated the fourth chapter to the definition of the main functional and non-functional requirements of our system. We first proceeded to the statement of some limitations of this scope, then we derived the major requirements for the system we wanted to develop. This mainly resulted in a list of elements to visualise in order to allow an appropriate investigation into quantitative usability data, as well as a generalised task model presenting all the functional requirements of the system.

Chapter 5 was the central chapter. Indeed, we tackled in it all the design issues related to the functional requirements exposed in Chapter 4. We first defined the data we needed in order to meet the visualisation requirements listed in the previous chapter. The organisation of this data led to the definition of two file formats: a configuration file containing the information common to all observations and an observation file containing all the information related to a particular observation.

Second, we tried to deal with the main design aspects of the interface. We started by introducing the concept of multiple view systems. This concept was crucial since we faced a wide range of data with multiple levels of complexity. Then, we derived some useful recommendations partly inspired from the design of a typical multiple view application: Spotfire. Among them, we can mention the simultaneous display of several views and the dynamic update of the views displayed when selecting and filtering information. Next, we designed each of the components of our interface separately and with regard to both the functional requirements from Chapter 4 and the recommendations coming from the Spotfire analysis. Finally, we dealt with the major sub-goal of this thesis by identifying a set of visualisation metaphors and discussing their design in order to find out the best ways to visualise the elements exposed in Chapter 4. The visualisation issue has been addressed with the design of 2D and 3D charts, tables as well as a video player.

The purpose of the sixth chapter was to discuss the implementation of the application. We first discussed the final sub-goal: the determination of the most appropriate implementation tool(s) to use. We opted for an application fully implemented in Java. Though Java was certainly not the only appropriate language, we showed in the first section of Chapter 6 that it was a perfectly valid implementation tool for data visualisation and that there was no fundamental objection to use it.

Then, we introduced the main elements helping to understand and maintain the code. We started with a diagram presenting the hierarchy of all interface-oriented components used with a reference to the corresponding Java classes, and we next gave the main programming conventions used within these classes.

To conclude, the last chapter carried out a global usability evaluation of the system we developed using a heuristic evaluation.

8.2 Future research

A first set of recommendations for future researches on this topic directly emerges from the limitations of the project's scope we adopted. Implementing a fully functional system is far from being easy and quick⁶ and we were therefore forced to limit its functional requirements with regard to the time we had at our disposal.

A first limitation we imposed is related to the data we collected. Contrary to what is available with Noldus, we did not consider any behavioral classes or modifiers for behaviors (cf. Sections 3.2.2.2 and 4.3.3). This restriction is quite strict as we sometimes do not have the necessary information on the behaviors to derive some basic measurements with accuracy. For example, when determining the time spent performing a particular behavior, we are forced to adopt conventions that are not always completely exact even if acceptable such as the fact that users cannot perform two behaviors at the same time, or the fact that a device related behavior always interrupts a continuous user related behavior being performed (at least for a while). Allowing more accuracy in the definition of the behaviors is thus probably the next step in improving our application.

A second limitation comes from our choice not to consider in detail the impact of the device nature on its usability. We only provided a few characteristics of the device to help the user in interpreting correctly the results of the usability analysis performed. However, the unprecedented growth in telecommunications we are facing nowadays lets us think that it is not reasonable to perform a usability test in the same way whatever the device that is used. An interesting topic for future research would then be to specialise the system to a particular device. Certain devices might indeed lead to consider other kinds of data and consequently new visualisation metaphors.

Another limitation involves the number of considered independent variables. We only took into account the five most often used independent variables (i.e. the gender, the age, the expertise and the language of the user) while it would have probably been better to let the user of the system determining their number and their nature.

A fourth restriction is the impossibility for the current application to access outside sources without having to encode manually the input files. The design of a usable interface with the most commonly used usability systems such as the Noldus Observer Video Pro is of the utmost importance and will have to be addressed in the future in order to make this system more easily usable for a majority of potential users.

At last, the evaluation of the system (Chapter 7) also stressed the need for a help function which cannot be ignored if one wants to help the user in one's task without referring to the documentation for each problem that could occur.

⁶ The whole application is currently more than 20000 lines long. Furthermore, we did not have at our disposal any tool to generate Java code automatically and this application has thus been fully manually implemented.

Other recommendations are related to the most appropriate ways to collect the usability data and to use the system. In other words, an investigation should be conducted into the different ways allowing the collection of data and the analysis of this data in order to discover usability problems efficiently.

A last recommendation concerns the methodology to use for the development of visualisation systems. We think that an evaluation and a validation of the methodology we followed could lead to the identification of guidelines for the design of visualisation systems.

Finally, though the system we developed seems to be very acceptable in a whole, a last issue that absolutely needs to be considered before any further researches is an evaluation of the aptitude of this system to fulfil expert users requirements both in terms of utility and usability. This evaluation should of course include a detailed analysis of the ability of the produced charts to catch the user's attention on potential usability problems within the collected usability data.

Appendices

Appendix A: Complete Extant Task Models

Complete Extant Task Model of the WebTrends Log Analyser.

1. Design a profile

1.1 Create a new profile

1.1.1 Create new profile

1.1.2 Edit new profile

1.1.2.1 Select profile

1.1.2.2 Edit profile

1.1.2.2.1 Edit Title, URL

1.1.2.2.1.1 Enter description

1.1.2.2.1.2 Enter log file URL

1.1.2.2.1.3 Enter path

1.1.2.2.1.4 Select log file format

1.1.2.2.2 Specify DNS lookup

1.1.2.2.2.1 Select DNS lookup methods

1.1.2.2.2.2 Select cache control options

1.1.2.2.2.3 Clear cache

1.1.2.2.3 Edit home page information

1.1.2.2.3.1 Enter home page file names

1.1.2.2.3.2 Enter website URL

1.1.2.2.3.3 Enter user name

1.1.2.2.3.4 Enter password

1.1.2.2.4 Specify filters

1.1.2.2.4.1 Add a new filter

1.1.2.2.4.1.1 Delete the Include Everything Filter

1.1.2.2.4.1.2 Add new filter

1.1.2.2.4.1.2.1 Select type

1.1.2.2.4.1.2.2 Enter filter's name

1.1.2.2.4.1.2.3 Select criteria for filtering

1.1.2.2.4.1.2.4 Confirm Add filter

1.1.2.2.4.2 Modify an existing filter

1.1.2.2.4.2.1 Select filter

1.1.2.2.4.2.2 Edit filter

1.1.2.2.4.2.2.1 Enter filter's name

1.1.2.2.4.2.2.2 Select criteria for filtering

1.1.2.2.4.2.2.3 Confirm edit filter

1.1.2.2.4.3 Delete an existing filter

1.1.2.2.4.3.1 Select filter

1.1.2.2.4.3.2 Delete filter

1.1.2.2.5 Activate FastTrends

1.1.2.2.5.1 Activate FastTrends

1.1.2.2.5.2 Maintain database

1.1.2.2.5.2.1 Select profile

1.1.2.2.5.2.2 Delete database entries affected by the changes

- 1.1.2.2.5.2.3 Confirm maintain database
 - 1.1.2.2.6 Activate Real-time analysis
 - 1.1.2.2.6.1 Activate FastTrends
 - 1.1.2.2.6.2 Activate Real-time analysis
 - 1.1.2.3 Confirm edit profile
 - 1.2 Modify an existing profile
 - 1.2.1 Select profile
 - 1.2.2 Edit profile
 - 1.3 Copy an existing profile
 - 1.3.1 Select profile
 - 1.3.2 Copy profile
 - 1.4 Delete an existing profile
 - 1.4.1 Select profile
 - 1.4.2 Delete profile
- 2. Save report settings**
 - 2.1 Select a profile
 - 2.2 Select to create report
 - 2.3 Select the data range for the report
 - 2.4 Select a file format for the report
 - 2.5 Select method and destination for saving report
 - 2.5.1 Select method
 - 2.5.2 Select destination and name
 - 2.6 Specify a style for the report
 - 2.6.1 Enter report title
 - 2.6.2 Select language
 - 2.6.3 Select style
 - 2.6.4 Select whether to include help cards in the report
 - 2.7 Specify the content of the report
 - 2.8 Memorise the report settings
 - 2.8.1 Enter a name for the report settings
 - 2.8.2 Confirm memorise the report settings
 - 2.9 Cancel create report
- 3. Customise reports**
 - 3.1 Customise styles
 - 3.1.1 Create a new style
 - 3.1.1.1 Create new style
 - 3.1.1.2 Select colours
 - 3.1.1.3 Select fonts
 - 3.1.1.4 Confirm create new style
 - 3.1.2 Modify an existing style
 - 3.1.2.1 Select style
 - 3.1.2.2 Select colours
 - 3.1.2.3 Select fonts
 - 3.1.2.4 Confirm modify existing style
 - 3.1.3 Delete an existing style
 - 3.1.3.1 Select style

- 3.1.3.2 Delete style
- 3.1.4 Rename an existing style
 - 3.1.4.1 Select style
 - 3.1.4.2 Rename style
 - 3.1.4.2.1 Enter style name
 - 3.1.4.2.2 Confirm rename style
- 3.2 Customise language
 - 3.2.1 Select language
 - 3.2.2 Confirm customise language

4. Run a report

- 4.1 Select profile
- 4.2 Create report
 - 4.2.1 Select a memorised report
 - 4.2.2 Run report

5. Schedule reports

- 5.1 Schedule events
 - 5.1.1 Schedule an event
 - 5.1.1.1 Add event
 - 5.1.1.2 Edit event
 - 5.1.1.2.1 Select profile
 - 5.1.1.2.2 Enter start date
 - 5.1.1.2.3 Enter start time
 - 5.1.1.2.4 Specify time between two consecutive analysis
 - 5.1.1.2.5 Select memorised report
 - 5.1.1.2.6 Select method for saving the report
 - 5.1.1.2.7 Select destination for saving report
 - 5.1.1.3 Specify an application to run before the scheduled event is processed
 - 5.1.1.3.1 Select file name of the program
 - 5.1.1.3.2 Select directory for the program to run
 - 5.1.1.3.3 Enter parameters for the program
 - 5.1.1.4 Specify an application to run after the scheduled event is processed
 - 5.1.1.4.1 Select file name of the program
 - 5.1.1.4.2 Select directory for the program to run
 - 5.1.1.4.3 Enter parameters for the program
 - 5.1.1.5 Select the processing priority
 - 5.1.2 Edit a scheduled event
 - 5.1.2.1 Select scheduled event
 - 5.1.2.2 Edit scheduled event
 - 5.1.3 Delete scheduled event
 - 5.1.3.1 Select scheduled event
 - 5.1.3.2 Delete scheduled event
- 5.2 View history of scheduled events at a given date
 - 5.2.1 Enter data
 - 5.2.2 View events

- 5.3 View the system's performance by event
 - 5.3.1 Select day
 - 5.3.2 View performances
- 5.4 Select options on how scheduled events are processed
- 5.5 Activate remote scheduling centre
 - 5.5.1 Specify a remote reporting server
 - 5.5.2 Start server
 - 5.5.3 Configure user account for remote scheduling
- 5.6 Set up scheduler to run as a service
 - 5.6.1 Enter username
 - 5.6.2 Enter password
 - 5.6.3 Start service
 - 5.6.4 Configure the service startup
 - 5.6.5 Add privileges to NT account

Complete Extant Task Model of the Noldus Observer Video Pro.

- 1 **Design a configuration**
 - 1.1 Specify the Data Collection Method
 - 1.1.1 Select the sampling method
 - 1.1.2 Select the number of actors
 - 1.2 Specify the Timing Method
 - 1.2.1 Select the duration
 - 1.2.1.1 Select Open Ended or
 - 1.2.1.2 Select Maximum
 - 1.2.1.2.1 Enter a maximum time
 - 1.2.1.2.2 Select the base
 - 1.2.2 Select the timing
 - 1.2.2.1 Select a resolution
 - 1.2.2.2 Enter a Sample Interval
 - 1.3 Specify Independent Variables
 - 1.3.1 Add an independent variable
 - 1.3.1.1 Enter a name
 - 1.3.1.2 Select data type
 - 1.3.1.2.1 Select Character or
 - 1.3.1.2.1.1 Enter an option
 - 1.3.1.2.1.2 Add the option
 - 1.3.1.2.2 Select Numeric
 - 1.3.1.2.2.1 Enter a minimum
 - 1.3.1.2.2.2 Enter a maximum
 - 1.3.2 Remove an independent variable
 - 1.3.2.1 Select an independent variable
 - 1.3.2.2 Remove it
 - 1.3.3 Replace an independent variable
 - 1.3.3.1 Select an independent variable
 - 1.3.3.2 Modify it
 - 1.3.3.3 Replace it
 - 1.4 Specify Subjects
 - 1.4.1 Add a subject
 - 1.4.1.1 Select the length of input code
 - 1.4.1.2 Enter the subject name
 - 1.4.1.3 Enter a code
 - 1.4.1.4 Enter a definition
 - 1.4.1.5 Add the subject
 - 1.4.2 Remove a subject
 - 1.4.2.1 Select a subject
 - 1.4.2.2 Remove it
 - 1.4.3 Replace a subject
 - 1.4.3.1 Select a subject
 - 1.4.3.2 Modify it
 - 1.4.3.3 Replace it

1.5 Specify Behavioral Classes

- 1.5.1 Add a behavioral class
 - 1.5.1.1 Select the length of input code
 - 1.5.1.2 Enter a behavioral class name
 - 1.5.1.3 Add the behavioral class
- 1.5.2 Remove a behavioral class
 - 1.5.2.1 Select a behavioral class
 - 1.5.2.2 Remove it
- 1.5.3 Replace a behavioral class
 - 1.5.3.1 Select a behavioral class
 - 1.5.3.2 Modify it
 - 1.5.3.3 Replace it
- 1.5.4 Specify Behavioral Elements
 - 1.5.4.1 Select a behavioral class
 - 1.5.4.2 View behavioral elements
 - 1.5.4.3 Modify behavioral elements
 - 1.5.4.3.1 Add a behavioral element
 - 1.5.4.3.1.1 Enter a name
 - 1.5.4.3.1.2 Enter a code
 - 1.5.4.3.1.3 Enter a definition
 - 1.5.4.3.1.4 Select a state
 - 1.5.4.3.1.5 Select a first modifier
 - 1.5.4.3.1.5.1 Select a second modifier
 - 1.5.4.3.1.6 Add the behavioral element
 - 1.5.4.3.2 Remove a behavioral element
 - 1.5.4.3.2.1 Select a behavioral element
 - 1.5.4.3.2.2 Remove it
 - 1.5.4.3.3 Replace a behavioral element
 - 1.5.4.3.3.1 Select a behavioral element
 - 1.5.4.3.3.2 Modify it
 - 1.5.4.3.3.3 Replace it

1.6 Specify Modifier Classes

- 1.6.1 Add a modifier class
 - 1.6.1.1 Enter a name
 - 1.6.1.2 Add the class
- 1.6.2 Remove a modifier class
 - 1.6.2.1 Select a modifier class
 - 1.6.2.2 Remove it
- 1.6.3 Replace a modifier class
 - 1.6.3.1 Select a modifier class
 - 1.6.3.2 Modify it
 - 1.6.3.3 Replace it
- 1.6.4 Specify Modifier Elements
 - 1.6.4.1 Add a modifier element
 - 1.6.4.1.1 Select the length of input code
 - 1.6.4.1.2 Enter a name
 - 1.6.4.1.3 Enter a code
 - 1.6.4.1.4 Enter a definition

- 1.6.4.1.5 Add the modifier element
- 1.6.4.2 Remove a modifier element
 - 1.6.4.2.1 Select a modifier element
 - 1.6.4.2.2 Remove it
- 1.6.4.3 Replace a modifier element
 - 1.6.4.3.1 Select a modifier element
 - 1.6.4.3.2 Modify it
 - 1.6.4.3.3 Replace it

1.7 Specify Channels

- 1.7.1 Add a channel
 - 1.7.1.1 Select focal subject
 - 1.7.1.2 Select an element class
 - 1.7.1.3 Add the channel
- 1.7.2 Remove a channel
 - 1.7.2.1 Select a channel
 - 1.7.2.2 Remove it
- 1.7.3 Replace a channel
 - 1.7.3.1 Select a channel
 - 1.7.3.2 Modify it
 - 1.7.3.3 Replace it

1.8 Review the configuration

1.9 Test the configuration

- 1.9.1 View the result

2. Collect Data

2.1 Customise the Event Recorder

- 2.1.1 Specify Operation
 - 2.1.1.1 Select frequency of edition of independent variables before observation
 - 2.1.1.2 Select frequency of edition of independent variables after observation
 - 2.1.1.3 Choose to get prompt for confirmation to end observation or
 - 2.1.1.4 Choose not to get prompt for confirmation to end observation
- 2.1.2 Customise windows
 - 2.1.2.1 Customise the Chechsheet
 - 2.1.2.1.1 Choose the optional columns to display
 - 2.1.2.1.2 Select the way to display score time
 - 2.1.2.1.3 Enter the column sizes
 - 2.1.2.2 Customise the Timers
 - 2.1.2.2.1 Choose the timers to display
 - 2.1.2.2.2 Select the time resolution
 - 2.1.2.3 Customise the Channel
 - 2.1.2.3.1 View Channel
 - 2.1.2.3.2 Enter the number of columns
 - 2.1.2.4 Customise the Checklist (not available)
 - 2.1.2.5 Customise the Codes
 - 2.1.2.5.1 Select the selection mode
 - 2.1.2.5.2 Select the visibility mode

- 2.1.2.5.3 Choose to sort or
 - 2.1.2.5.3.1 Select the sort criteria
- 2.1.2.5.4 Choose not to sort
- 2.1.2.6 Customise the System Status
 - 2.1.2.6.1 Choose the information to display
- 2.1.2.7 Customise the Tool base
 - 2.1.2.7.1 Select the appearance
 - 2.1.2.7.2 Select the icon size
 - 2.1.2.7.3 Select an icon
- 2.1.2.8 Customise Notepad
 - 2.1.2.8.1 Choose to add event string to timestamp or
 - 2.1.2.8.2 Choose not to
- 2.1.3 Customise Data Entry
 - 2.1.3.1 Select the response to invalid input code
 - 2.1.3.2 Select the response to double states
 - 2.1.3.3 Choose data entry options
 - 2.1.3.4 Select the order of data entry
 - 2.1.3.5 Select the key for missing Actor/Modifier
- 2.1.4 Customise Data Storage
 - 2.1.4.1 Select the file name type
 - 2.1.4.2 Select the file format
 - 2.1.4.3 Choose the safety options
 - 2.1.4.3.1 Enter a required minimum amount of free disk space
 - 2.1.4.3.2 Choose to auto-save data or
 - 2.1.4.3.2.1 Select the auto-save data frequency
 - 2.1.4.3.3 Choose not to auto-save data
- 2.1.5 Customise the Auditory Feedback
 - 2.1.5.1 Select the auditory feedback
 - 2.1.5.2 Choose signal at sample or
 - 2.1.5.3 Choose no signal at sample
- 2.2 Define independent variables
 - 2.2.1.1 Enter a surname
 - 2.2.1.2 Enter a first name
 - 2.2.1.3 Select a level
 - 2.2.1.4 Select a home language
 - 2.2.1.5 Enter an age
- 2.3 Perform an observation session
 - 2.3.1 Start an observation
 - 2.3.1.1 Enter a description of the observational data file
 - 2.3.1.1.1 Enter a name
 - 2.3.1.1.2 Enter a title
 - 2.3.1.2 Enter independent variables
 - 2.3.1.2.1 Enter the name of the test person
 - 2.3.1.2.2 Enter the program version
 - 2.3.1.2.3 Enter the name of the software product tested
 - 2.3.1.3 Initialise channels
 - 2.3.1.3.1 Initialise the channels' behaviors
 - 2.3.1.4 Start the observation

- 2.3.2 Edit observational data
 - 2.3.2.1 View observational data recorded at that time of the observation
 - 2.3.2.2 Edit it
- 2.3.3 Suspend the observation
 - 2.3.3.1 Modify the current channels' behaviors
 - 2.3.3.2 Resume the observation
- 2.3.4 Log events
 - 2.3.4.1 Select a user
 - 2.3.4.2 Select a behavior
- 2.3.5 Scan channels (States or Events)
 - 2.3.5.1 View channels
 - 2.3.5.2 Modify the current channels' behaviors
 - 2.3.5.3 Resume the observation

3. Data Analysis using a Time-Event Plot

- 3.1 Fill cells in the worksheet
 - 3.1.1 Fill cell by cell
 - 3.1.1.1 Select a cell
 - 3.1.1.2 View list
 - 3.1.1.3 Select an element of the list
 - 3.1.2 Fill consecutive cells of the first column
 - 3.1.2.1 Select the cells to fill
 - 3.1.2.2 View list
 - 3.1.2.3 Select an element of the list
 - 3.1.3 Fill most cells of a trace
 - 3.1.3.1 Select a cell
 - 3.1.3.2 Plot a channel
- 3.2 View graph
 - 3.2.1 Fill cells in the worksheet
 - 3.2.2 Create graph
 - 3.2.3 View graph
- 3.3 Zoom in
 - 3.3.1 Select the Zoom in feature
 - 3.3.2 Select one end of the graph section to view
 - 3.3.3 Select the other end
- 3.4 Zoom out
 - 3.4.1 Select the Zoom out feature
- 3.5 View the full view
 - 3.5.1 Select the full view feature
- 3.6 Define a time-based filter
 - 3.6.1 Select the point from where to plot the graph
 - 3.6.1.1 Select start of observation or
 - 3.6.1.2 Select event or
 - 3.6.1.2.1 Select a member of the action list
 - 3.6.1.2.2 Select a member of the behavior list
 - 3.6.1.2.3 Select a member of the modifier 1 list
 - 3.6.1.2.4 Select a member of the modifier 2 list
 - 3.6.1.3 Select time

- 3.6.1.3.1 Enter a time
- 3.6.2 Select the point to where to plot the graph
 - 3.6.2.1 Select end of observation or
 - 3.6.2.2 Select event or
 - 3.6.2.2.1 Select a member of the action list
 - 3.6.2.2.2 Select a member of the behavior list
 - 3.6.2.2.3 Select a member of the modifier 1 list
 - 3.6.2.2.4 Select a member of the modifier 2 list
 - 3.6.2.3 Select time
 - 3.6.2.3.1 Enter a time
- 3.7 Plot a channel
 - 3.7.1 Select a channel
 - 3.7.2 Select a subject
- 3.8 Specify the grid
 - 3.8.1 Choose to display the grid or
 - 3.8.1.1 Select the grid layout
 - 3.8.2 Choose not to display the grid
- 3.9 Specify the legend
 - 3.9.1 Choose to display the legend or
 - 3.9.1.1 Select the save option
 - 3.9.2 Choose not to display the legend
- 3.10 Specify the title
 - 3.10.1 Choose to display the title or
 - 3.10.2 Choose not to display the title
 - 3.10.3 Enter a title
- 3.11 Select the trace styler

Appendix B: Class Diagram Notation

Class Diagram Notation

For a more detailed description of the class diagram notations, see [Baldwin; UML].

Class

A class is an abstract type characterised by properties (attributes and methods) common to a set of objects and allowing the creation of objects having those properties. It is thus usually represented as a rectangle typically divided into three regions, one for each of those properties. A class can be described as a template from which objects are instantiated (objects show instances of classes, and are displayed in objects diagram).

Example:

Car	Class name
Owner Brand Colour	Class attributes
Start Drive Stop	Class methods

We can notice that displaying a class without representing the attributes or the methods does not mean that this class has none of them. It is just a visual filter, aimed at giving the model more abstraction.

Class Diagram

A class diagram is set of static elements (like classes) that show the structure of a model. A class diagram provides an overview of a system by showing its classes and the relationships among them. Class diagrams are static - they display what interacts, but not what happens when they do interact: they thus ignore the temporal and dynamic aspects.

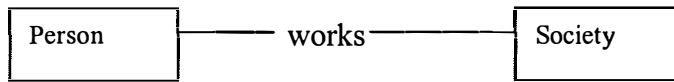
Forms of class association

Classes may be associated with each other. We consider four forms of association:

- Association
- Aggregation
- Composition
- Inheritance

Association:

An association expresses a bi-directional connection between two classes.



Aggregation:

Aggregation is a form of asymmetric association that specifies a whole-part relationship between the aggregate (whole) class and a subordinate (part) class. An aggregation thus expresses a string coupling and a subordination relationship. It can be seen as a set/element relationship.

An aggregation may among others express that:

- A class is part of another
- An action on a class supposes an action on another one
- A change in the state of a class supposes a change in another class state

An instance of an aggregated element may exist without the aggregate, and the contrary is true too: their respective lifecycles may be independent.

An aggregation relationship may exist, for instance, between an E-mail class (the aggregate) and a Text class (a subordinate), and maybe an Attached file class (another subordinate). The Text and File classes can obviously exist independently from the E-mail class. There is no physical relationship between them.

The aggregation symbol:



Composition:

Composition is a form of symmetric association that specifies a whole-part relationship between the composition (whole) class and a subordinate (part) class in which removing the whole also removes the parts. In relational databases, a cascading delete is a good example of a composition relationship. This thus means that the elements' lifecycles are tied. A composition can be seen as a strong aggregation (aggregation by value).

A composition relationship may exist between, for instance, a Person class and a Head class.

The composition symbol:

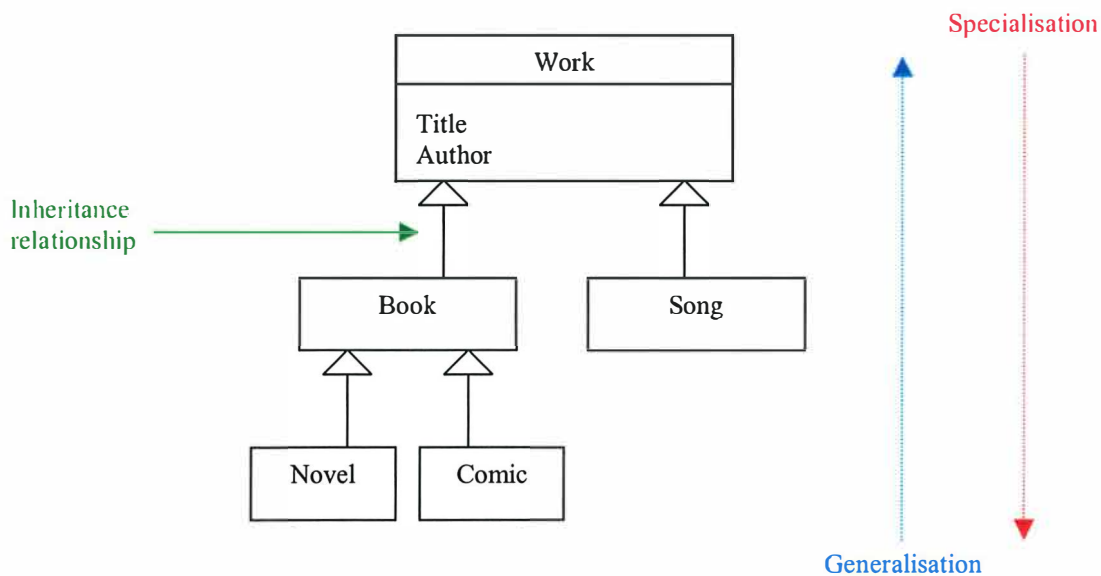


Inheritance:

Inheritance is used to build class hierarchies. Two opposite and complementary kinds of relationship exist that allows one to order the classes:

- Specialisation: descending mechanism which consists in extending the properties of a class, under the shape of sub-classes, that are more specific
- Generalisation: ascending mechanism, which consists in factorising the properties of a set of classes, under the shape of a super-class, more abstract

Example:

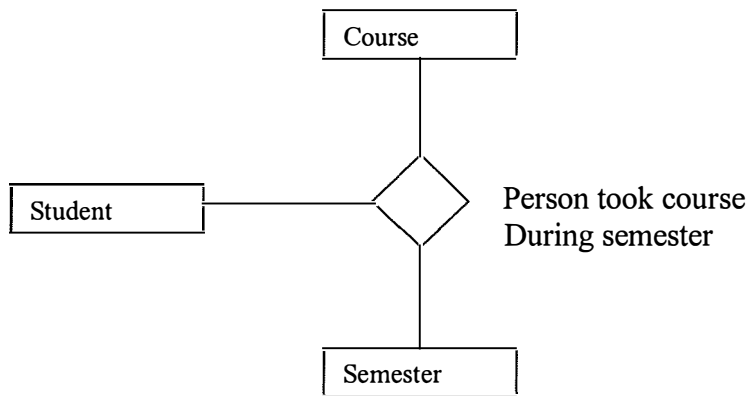


OMT notation

We have used a few standards defined in the OMT notation in some of the presented diagrams. Therefore we present quickly the basic notations of this standard that we have included in the concerned diagrams. For more information, see [Chinodom; Henocque].

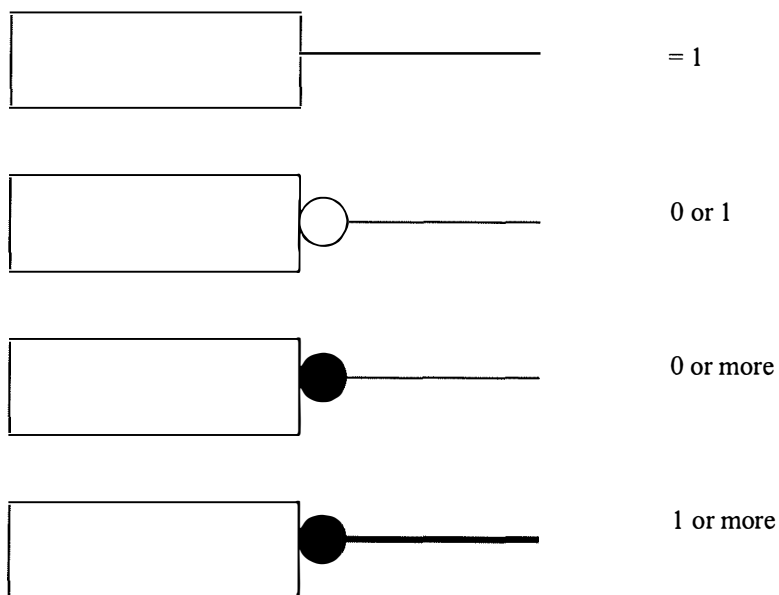
Ternary association

The diagram shows an example of ternary association in OMT notation.



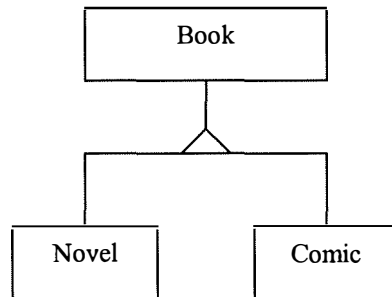
Cardinality of associations

The cardinality determines the number of instances of a class that can be tied to an associated class instance. The diagram below shows examples of “cardinality associations” in the OMT notation.



Inheritance relationship

With the OMT notation, an inheritance relationship takes the following shape:



Appendix C: BNF notations for the file formats

Configuration file

**<file> ::= <list_of_behaviors> [<optional_sequence_of_behaviors>]
<list_of_observations>;**

**<list_of_behaviors> ::= LIST OF BEHAVIORS: <space> <crLf> USER
BEHAVIORS: <space> <crLf> <behavior> {<behavior>} DEVICE
BEHAVIORS: <behavior> {<behavior>;}**

<behavior> ::=
 Name: <space> <behavior_name> <space> <crLf>
 Description: <space> <string> <space> <crLf>
 Type: <space> <type> <space> <crLf>;

<type> ::= discrete | continuous;

<behavior_name> := <identifier>;

**<optional_sequence_of_behaviors> ::= OPTIMAL SEQUENCE OF BEHAVIORS:
<space> <crLf> <behavior_name> <space> <crLf> {<behavior_name>
<space> <crLf>} [<media>];**

**<list_of_observations> ::= LIST OF OBSERVATIONS: <space> <crLf>
<observation> {<observation>;}**

<observation> ::= <location> <space> <crLf>;

<media> ::= MEDIA: <space> <location> <space> <crLf>;

<location> ::= ".\" <path> <file_name> | ":\\" <path> <file_name>;

<path> ::= {<string> "\"};

<file_name> ::= <string> "." <extension>;

<extension> ::= dof;

<identifier> ::= <alpha> <alpha_digit> {1..9};

<alpha_digit> ::= <alpha> | <digit>;

**<alpha> ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" |
"k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" |
"w" | "x" | "y" | "z" | "A" | "B" | "C" | "D" | "E" | "F" | "G" |
"H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
"S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z";**

<digit> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0";

<string> ::= <char> {<char>;}

```
<char> ::= <alpha_digit> | " " | "<" | "!" | "#" | "$" | "&" | "" | "(" |
")" | "*" | "+" | "," | "-" | "." | "/" | ":" | ";" | "<" | "=" | ">" |
"?" | "@" | "[" | "]" | "^" | "_" | "`" | "{" | "|" | "}" | "~" | "%";
```

```
<crlf> ::= <cl> {<cl>}
```

```
<cl> ::= ASCII character %xD0 for the end of line
```

```
<space> ::= {" "};
```

Constraints:

- Each **identifier** in the **list_of_behaviors** is unique.
- Each observation file name in the **list_of_observations** is unique, i.e. two observations in the configuration file must have different file names, even if they are stored in different directories.
- Each **behavior_name** must correspond to the name of one of the behaviors defined in the **list_of_behaviors**.

Observation file

We represent the observation file format for the data using the BNF notation as follows:

```
<file> ::= <observation_attributes> [<media>] <subjects> <observation>;
```

```
<observation_attributes> ::=
OBSERVATION NAME: <space> <char>{1..20} <space> <crlf>
OBSERVATION DESCRIPTION: <space> <string> <space>
<crlf>
START DATE:
<space> <2digits> <2digits> "-" <2digits> "-" <2digits> <space>
<crlf>
START TIME:
<space> <2digits> ":" <2digits> ":" <2digits> "." <2digits>
<crlf>;
TOTAL TIME: <space> {<digits>} "." <2digits> <crlf>;
TASK COMPLETED: <space> <yes_no> <crlf>;
```

```
<yes_no> ::= yes | no;
```

```
<media> ::= MEDIA: <space> <location> <space> <crlf>;
<location> ::= "." <path> <file_name> | ":" <path> <file_name>;
<path> ::= {<string> "\"};
<file_name> ::= <string> "." <extension>;
<extension> ::= mpg | avi;
```

```

<subjects> ::= SUBJECTS: <space> <crLf> <subject>;
<subject> ::= <user> <device>;
<user> ::=
    USER DESCRIPTION: <space> <space> <crLf>
    Surname: <space> <string> <space> <crLf>
    Name: <space> <string> <space> <crLf>
    Gender: <space> <gender> <space> <crLf>
    Age: <space> <age> <space> <crLf>
    Language: <space> <string> <crLf>
    Expertise: <space> <expertise> <space> <crLf>;

```

```

<device> ::=
    DEVICE DESCRIPTION: <space> <crLf>
    Type: <space> <string> <space> <crLf>;
    Media Available: <space> <string> <space> <crLf>;
    Screen Size: <space> <string> <space> <crLf>;
    Colours: <space> <string> <space> <crLf>;
    Interaction Tools: <space> <string> <space> <crLf>;
    Other features: <space> <string> <space> <crLf>;

```

```

<gender> ::= male | female;
<age> ::= <digit> [<digit>]
<expertise> ::= novice | medium | expert;

```

```

<observation> ::=
    OBSERVATION START <space> <crLf> <observation_list>
    OBSERVATION END <space> <crLf>;

```

```

<observation_list> ::=
    {<space> <time> <space> <behavior_name> <space> <crLf>};
<time> ::= <digit> {<digit>} "." <2digits>;
<subject> ::= User | Device;
<behavior_name> ::= <identifier>;

```

```

<identifier> ::= <alpha> <alpha_digit>{1..9};

```

```

<alpha_digit> := <alpha> | <digit>;

```

```

<alpha> ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" |
"k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" |
"w" | "x" | "y" | "z" | "A" | "B" | "C" | "D" | "E" | "F" | "G" |
"H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
"S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z";

```

<digit> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0";

<2digit> ::= **<digit>** {**<digit>**};

<string> ::= **<char>** {**<char>**};

<char> ::= **<alpha>** | **<digit>** | " " | "<" | "!" | "#" | "\$" | "&" | "" |
"(" | ")" | "*" | "+" | "," | "-" | "." | ":" | ";" | "<" | "=" | ">" |
"?" | "@" | "[" | "]" | "^" | "_" | "`" | "{" | "|" | "}" | "~" | "%";

<crlf> ::= **<cl>** {**<cl>**}

<cl> ::= ASCII character %xD0 for the end of line

<space> ::= {" "}

Constraints:

- The **time** in the **observation_list** starts at 0.00.
- A **behavior_name** in the **observation_list** corresponds to a "Name" of behavior in the **list_of_behaviors** of the Configuration File.
- The **observation-list** is sorted on time.

Appendix D: Spotfire Test File

The complete test file used to analyse Spotfire DecisionSite.

Observation	Time	Student Behaviors	Computer Behaviors
1	0	Read	
1	0		No
1	0		NoC
1	5.28		Screen1
1	5.44	Key	
1	17.44		Screen2
1	17.64	Key	
1	22.72		Screen3
1	37.2	Read	
1	38.36		Reminder
1	38.36	Key	
1	38.68		No
1	39.4	Read	
1	48.04	Key	
1	48.72		Screen4
1	50.2	Read	
1	56.76		Screen5
1	57.48	Key	
1	57.52		Screen6
1	58.2	Read	
1	59.96	Key	
1	88.96		YesC
1	90.88		NoC
1	91.88	Read	
1	91.92		Screen7
1	96.72	Key	
1	98.88		Screen8
1	99.92	Read	
1	102.84	Key	
1	118.36		YesC
1	118.96		NoC
1	119.96	Read	
1	120.2		Screen9
1	129.44	Key	
1	130.28		Screen10
1	130.84	Read	
1	141.4	Key	
1	141.72		Screen11
1	142.36	Read	
1	150.12	Key	
2	0	Read	
2	3.8	Key	
2	3.24		Screen1
2	5.56	Read	
2	14.16	Key	
2	14.32		Screen2

2	15.08 Read	
2	28.88 Key	
2	19.44	Screen3
2	29.72 Read	
2	30.2	Reminder
2	30.52 Key	
2	37.56	No
2	37.84	Screen4
2	38.38 Read	
2	42	Reminder
2	42.4	No
2	50.02 Key	
2	51.16	Screen5
2	73.12 Read	
2	74.04	Screen6
2	74.76 Key	
2	75.92	YesC
2	78.84	Reminder
2	79.28 Read	
2	79.8 Key	
2	80.68	NoC
2	81	No
2	82 Read	
2	84.4	Screen7
2	84.76 Key	
2	86 Read	
2	88.56	Screen8
2	104.4 Key	
2	104.6	YesC
2	105.04	NoC
2	105.72 Key	
2	105.92	Screen9
2	106.6 Read	
2	115.6 Key	
3	0 Read	
3	0.5	Screen1
3	2.39 Key	
3	2.78	Screen2
3	3.02 Read	
3	4 Key	
3	6.54	YesC
3	7.01	NoC
3	7.56 Read	
3	8.3	Screen3
3	9.23 Key	
3	9.45	Screen4
3	9.85 Read	
3	10.02 Key	
3	11	YesC
3	12.36	NoC
3	12.98 Read	
3	13.24	Screen5
3	13.35 Key	

3	14.52	Screen6
3	15.2 Read	
3	19.3 Key	
3	20.53	Screen7
3	21.32 Read	
3	22.67	Reminder
3	35.68	No
3	42.11 Key	
3	46.31	Screen8
3	47.96 Read	
3	56.2	Screen9
3	57.26 Key	
3	69.3	YesC
3	72.54	Reminder
3	76.58 Read	
3	81.31 Key	
3	86.5	NoC
3	87.03	No
3	96.44	Read
3	98.5	Screen10
3	100.25 Key	

Bibliography

- [Bachman, 1989] Bachman R. D., *A Methodology for comparing the software interfaces of competitive products*. Proc. Human Factors Society 33rd Annual Meeting, 1989.
- [Bailey, 1993] Bailey G., *Iterative Methodology and Designer Training in Human-Computer Interface Design*, PROC. ACM INTERCHI'93 Conf. (Amsterdam), 1993.
- [Baldonado et al., 2000] Baldonado M. Q. W., Woodruff A. and Kuchinsky A., *Guidelines for Using Multiple Views in Information Visualization*, 2000.
- [Baldwin] Baldwin D., *UML Tutorial: Unified Modelling language – Class & Object Diagrams*,
<http://www.auldenfire.com/aitpncc/resources/uml.shtml>
- [Beaton] Beaton R.J., *Human Information Processing, Chapter 4: Spatial Displays*,
http://filebox.vt.edu/users/yogeshdb/Human%20Information%20Processing/L04-Spatial_Displays.ppt
- [Benel et al., 1991] Benel D. C. R., Ottens D. and Horst R., *Use of an eyetracking system in the usability laboratory*. Proc. Human Factors Society 35th Annual Meeting, 461-465, 1991.
- [Bevan et al. 1991] Bevan N., Kirakowski J. and Maissel J., Proceedings of the 4th International Conference on HCI, Stuttgart, September 1991.
- [Brooke et al, 1990] Brooke J, Bevan N, Brigham F, Harker S, Youmans D (1990). *Usability statements and standardisation - work in progress in ISO*. In: Human Computer Interaction - INTERACT'90, D Diaper et al (ed), Elsevier.
- [Brown, 1995] Brown J. R., Earnshaw R., Jern M. and Vince J., *Visualization Using Computer Graphics To Explore Data And Present Information*, 1995.
- [Byrne, 1989] Byrne J. G.: *Competitive Evaluation in Industry: Some Comments*. Proc. Human Factors Society 33rd Annual Meeting, 1989.
- [Card et al., 1999] Card S. K., MacKinlay, J.D. and Shneiderman B. (eds): *Readings in Information Visualization: Using Vision to Think*. San Francisco, California, Morgan Kaufmann Publishers, Inc., 1999.
- [Caroll and Rosson, 1987] Caroll J. M. and Rosson M. B., *The Paradox of the Active User, in Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, edited by Caroll J. M., Cambridge: Bradford Books/MIT Press, 1987.

- [Chinodom] Chinodom S., *Comparison of Rumbaugh and Booch Notations*, in Object Oriented Application Development, Chapter 13, Burapha University, Computer Science Department, Thailand, <http://www.compsci.buu.ac.th/~seree/lecture/ooad/Chap13.pdf>
- [Darville and Van Espen, 2000] Darville C. and Van Espen S., *Study of the Virtual Reality Technology used in the Visualization of Business Information*. Masters thesis. Institut d'Informatique, FUNDP Namur, Belgium, 2000.
- [Diaper, 1989] Diaper D., *Task Observation for Human-Computer Interaction*, In Diaper D., *Task Analysis for Human-Computer Interaction*. Ellis Horwood, Chichester, U.K., 1989.
- [Dix and Finlay, 1998] A. Dix and J. Finlay, *Human-Computer Interaction*, Second Edition, 1998.
- [Eason, 1988] *Information Technology and Organisational Change*, Taylor and Francis, 1988.
- [El ansari and Vanbrabant, 2001] El Ansari A. and Vanbrabant A., *Interactive Visualization of Large Data Sets with the Java and VRML Technology*. Masters thesis, Institut d'informatique, FUNDP, Namur, Belgium.
- [EIA] The US Energy Information Administration web site, by the Official Energy Statistics from the US government, <http://www.eia.doe.gov/>
- [Galitz and Wilbert, 1989] Galitz and Wilbert, O. *Handbook of screen Format Design*, 3d ed. Wellesey, MA: QED Information Sciences, 1989.
- [Fowler, 1995] Fowler S. L. and Stanwick V. R., *The GUI Style Guide*, AP Professional, 1995.
- [Good et al., 1986] Good M., Spine T. M., Whiteside J. and George P. (1986): *User-derived impact analysis as a tool for usability engineering*, Proc. ACM CHI'86 Conf. (Boston, MZ, 13-17 April), 241-246.
- [Gould and Lewis, 1985] Gould J. D. and Lewis C. H., *Designing for Usability: Key Principles and What Designers Think*. Communications of the ACM 28, 3, March 1985.
- [Grief, 1991] Grief S., *Organisational Issues and Task Analysis*, in Schakel B. and Richardson S., *Human Factors for Informatics Usability*, Cambridge University Press, Cambridge, 1991.
- [Grudin, 1992] Grudin J., *Utility and Usability, Research Issues and development contexts*. *Interacting with computers* 4, 2, August 1992.

- [Harrison, 1991] Harrison B. L., *Video Annotation and multimedia interfaces: From Theory to Practice*, Proc. Human Factors Society Thirty-Fifth Annual Meeting, 1991.
- [Henocque] *Cours de Génie Logiciel: Introduction à la méthode OMT*, Ecole supérieure d'ingénieurs de Luminy,
<http://www.esil.univ-mrs.fr/~henocque/polyomt.pdf>
- [Hewett and Scott, 1987] Hewett T. T. and Scott S., *The Use of Thinking-out-loud and Protocol Analysis in Development of a process model of Interactive Database Searching*, Proc. IFIP INTERACT'87 Second Intl. Conf. Human-Computer Interaction (Stuttgart, Germany), 1987.
- [Hong et al.] Hong J. I., Heer J., Waterson S. and Landay J. A., *WebQuilt: A Proxy-based Approach to Remote Web Usability Testing*, Group for User Interface Research, Computer Science Division, University of California at Berkeley,
<http://guir.berkeley.edu/projects/webquilt/pubs/acmTOIS-webquilt-final.htm>
- [ISO ,1991b] ISO (1991b) *Software product evaluation - Quality characteristics and guidelines for their use*, ISO DIS 9126.
- [ISO 9241, 1998] ISO 9241-11 : *Guidance on Usability*, in ISO 9241, 1998.
- [Java, 2000] Gosling J., Joy B., Steele G. and Brocha G., *The Java Language Specification*, Sun Microsystems, Inc., 2000.
- [Kensing and Munk-Madsen, 1993] kensing F. and Munk-Madsen A., PD: *Structure of the Toolbox*, Communications of the ACM 36, 4, 1993.
- [Lanning et al. 2000] Lanning T., Wittenburg K., Fyock C. and Li G., *Multidimensional Information Visualization through Sliding Rods*, 2000.
- [Lewis, 1982] Lewis C., *Using the "Thinking-Aloud" Method in Cognitive Interface Design*, IBM Research Report RC 9265, IBM Thomas J. Watson Research Centre, Yorktown Heights, NY, 1982.
- [Lindgaard, 1991] Lindgaard G., *Impressions from HUSAT*. CHISIG Newsletter (Computer-Human Interaction Special Interest Group of the Ergonomics Society of Australia), August 1991.
- [Lund, 1985] Lund M. A., *Evaluating the User Interfaces: The Candid Camera Approach*, Proc. CHI' 85-Human Factors in Computing Systems, ACM, New York, 1985.

- [Mack and Burdett, 1992] Mack R. L., and Burdett J. M., *When Novices elicit knowledge: Question-Asking in Designing, Evaluating and Learning to Use Software*, in Hoffman R., *The Psychology of Expertise: Cognitive Research and Empirical AI*, Springer-Verlag, New-York, 1992.
- [Macleod and Rengger, 1993] Macleod M. and Rengger R.: *The development of DRUM: a software tool for video-assisted usability evaluation*, In: JL Alty et al. (Eds.) *People and Computers VIII (Proc. of HCI'93 Conf., Loughborough UK)*. Cambridge: CUP, 293-309, 1993.
- [Macleod et al., 1994] Macleod M., Bowden R. and Bevan N.: *The MUSiC Performance Measurement Method*, in HCI '96, Tutorial 14, *Measuring Usability - MUSiC Methods*, (Bevan N.). London: The British HCI Group, 1994.
- [Microsoft, 1992] Microsoft Corporation. *The Windows Interface: An application Design Guide*, Redmond, WA: Microsoft Press, 1992.
- [Moran and Carroll, 1994] Moran T. P. and Carroll J. M., *Design Rationale*, Lawrence Erlbaum Associates, Hillsdale, NJ.
- [Munzner, 2000] Munzner T.: *Interactive Visualization of Large Graphs and Networks*. PhD thesis. Stanford University, 2000, http://graphics.stanford.edu/papers/munzner_thesis/
- [Newman, 1995] Newman W. M. and Norman D. A., *Interactive System Design*, Addison-Wesley, 1995.
- [Nielsen, 1993] Nielsen J., *Usability Engineering*, Academic Press Limited, 1993.
- [Nielsen and Mack, 1994] Nielsen J. and Mack R. L., *Usability Inspection Methods*, John Wiley & Sons, Inc., New York, 1993.
- [Nielsen et al, 1993] Nielsen J., Desurvivre H., Kerr R., Rosenberg D., Salomon G., Molich R. and Steward T., *Comparative design review: An Exercise in Parallel Design*, Proc. ACM INTERCHI'93 Conf. (Amsterdam, The Netherlands), 1993.
- [Nielsen et al, 1994] Nielsen J., Fernandes T., Wagner A., Wolf R. and Ehrlich K., *Diversified Parallel Design: Contrasting Design Approaches*, ACM CHI'94 Conference Companion (Boston), 1994.
- [Noirhomme, 2000] Noirhomme M., Nahimana A. and De Greift B., *Graphic Library: Detailed Functional Analysis for Temporal Star, Simple Star and Urchin*, ISO-3D Project, 2000.
- [Noldus] The Noldus website, www.noldus.com.

- [North, 2000] Chris North and Ben Schneiderman, *Snap-Together Visualization: A user Interface for Coordinating Visualizations via Relational Schemata*, 2000.
- [Perlman, 1989] Perlman G., *Coordinating Consistency of User Interfaces, Code, Online Help, and Documentation with Multilingual/Multitarget Software Specification*. In Nielsen J., *Coordinating User Interfaces for Consistency*, Academic Press, Boston, 1989.
- [Root and Draper, 1983] Root R. W. and Draper S., *Questionnaires as a Software Evaluation Tool*, Proc. ACM CHI'83 Conf. (Boston, MA, 12-15 December), 1983.
- [Rubin, 1994] Rubin J., *Handbook of Usability Testing: How to Plan, Design and Conduct Effective Tests*, John Wiley & Sons, Inc., 1994.
- [Schmidt, 1988] Schmidt K., *Functional analysis instrument*. In Schaefer G., Hirschheim R., Harper M., Hansjee R., Domke M., and Bjorn-Andersen N., *Functional Analysis of Office Requirements: A Multiperspective Approach*. Wiley, Chichester, U.K. 261-289.
- [Schneiderman, 1998] Schneiderman B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Third Edition, Addison Wesley Longman, Inc., 1998.
- [Spotfire] Spotfire website, <http://www.spotfire.com>
- [Swallow et al.] Swallow J., Hameluck D. and Carey T., *User Interface Instrumentation for Usability Analysis: A Case Study*, <http://watserv1.uwaterloo.ca/~tcarey/casestudy.html>
- [Thomson] Thomson K., *Environments for collecting usability data*, The University of Waikato, New Zealand, <http://www.cs.waikato.ac.nz/~kthomson/HomePage/collecting.html>
- [Tufte, 1983] Tufte E. R., *The Visual Display of Quantitative Information*, Cheshire, CT: Graphics Press, 1983.
- [Tufte, 1990] Tufte E. R., *Envisioning Information*, Cheshire, CT: Graphics Press, 1990.
- [Tufte, 1993] Tufte E. R., *Workshop on Presenting Data and Information*, Arlington, VA, July 28, 1993.
- [UML] *UML en français*, <http://uml.free.fr>
- [WebTrends] The WebTrends website, www.webtrends.com

- [Wharton et al., 1992] Wharton C., Bradford J., Jeffries R. and Frankze M., *Applying Cognitive Walkthroughs to More Complex User Interfaces: Experiences, Issues, and Recommendations*. Proceedings ACM CHI'92 Conference (Monterey, CA, May), 1992.
- [Whiteside et al., 1988] Whiteside, J., Bennett, J. L. and Holtzblatt, K. *Usability Engineering: Our Experience and Evolution*, in Helander, M. (Ed.) Handbook of Human-Computer Interaction. Elsevier Science Publishers, 1988.
- [Wilson, 1997] Wilson C., *Usability Techniques: Analysing and Reporting Usability Data*, Usability Interface, October 1997
<http://www.stcsig.org/usability/newsletter/9710-analyzing-data.html>